THE OHIO STATE UNIVERSITY

High-Level Connectionist Models

Jordan B. Pollack
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science

Department of the Navy
Office of Naval Research
Arlington, Virginia 22217

Grant No. N00014-89-J-1200
Semiannual Report

August 1989

THE OHIO STATE UNIVERSITY

High-Level Connectionist Models

Jordan B. Pollack
Laboratory for Artificial Intelligence Research
Department of Computer and Information Science

Department of the Navy
Office of Naval Research
Arlington, Virginia 22217

Grant No. N00014-89-J-1200
Semiannual Report
RF Project 767172/721651

August 1989

High-Level Connectionist Models
ONR N00014-89-J-1200
Semi-Annual Progress Report
July 1989

Jordan B. Pollack
Laboratory for AI Research
Computer & Information Science Dept.
Ohio State University
Columbus, OH 43210
pollack@cis.ohio-state.edu

The major achievement of this first semiannum was the significant revision and extension of the Recursive Auto-Associative Memory (RAAM) work for publication in the journal *Artificial Intelligence*. Included as an appendix to this report, the article includes several new elements:

1) Background —

The work was more clearly set into the area of distributed representations, machine learning, and the adequacy of the connectionist approach for high-level cognitive modeling;

2) New Experiment—

RAAM was applied to finding compact representations for sequences of letters;

3) Analysis —

The developed representations were analyzed as features which range from categorical to distinctive. Categorical features distinguish between conceptual categories while distinctive features vary within categories and discriminate or label the members. The representations were also analyzed geometrically;

4) Applications —

Feasibility studies were performed and described on inference by association, and on using RAAM-generated patterns along with cascaded networks for natural language parsing. Both of these remain long-term goals of the project.

There are several other areas that are currently being explored, and which should be written up in the second semiannum:

Discrete Analog Systems

One problem for most recurrent or sequential work in connectionism is the default assumption of real arithmetic implemented in floating point. This means that states (or internal representations) are very imprecise, as there is no equality test. We have been experimenting with an activation function based upon the inverse of Cantor's function, shown below, which is a sigmoid-shaped step-function, and have been

able to use standard neural network learning algorithms with it. One result so far is a RAAM which *exactly* reconstructs its trees.

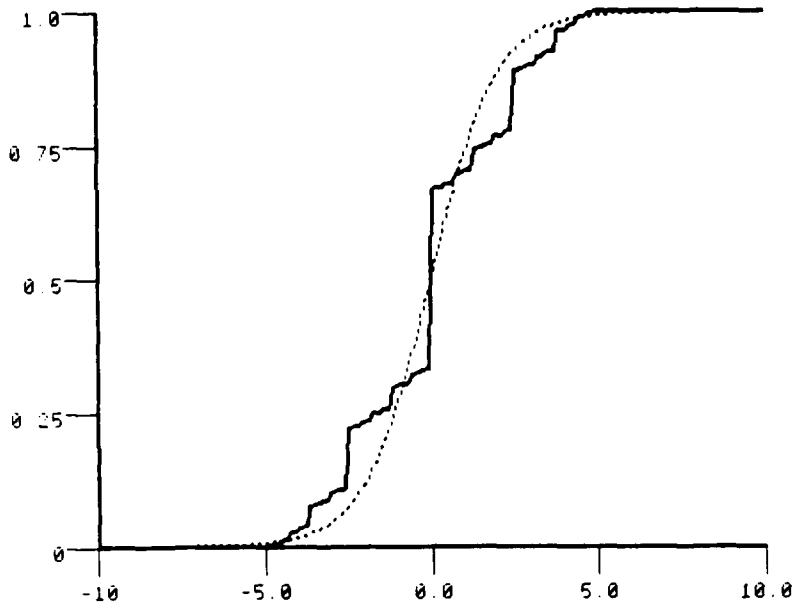

Figure 1. Cantor's sigmoid

Inductive Inference

J. Feldman set out (on an electronic bulletin board) the problem of inductive inference of finite state automata from language examples as a possible benchmark for connectionist networks. This is now a very active area, with research ongoing at CMU, Toronto, and UMass. Sequential Cascaded Networks had already shown some promise in this area, on the parity and balanced parenthesis languages. With a simple modification, they have worked on more complex test cases (from a 1982 paper by M. Tomita).

Chaotic Behavior

One of problems that plague modern connectionist learning algorithms is that gradient descent is susceptible to local minima. This has been discounted by the originators of Back-Propagation, but it is generally known that "sometimes it converges and sometimes it doesn't." It is also known that if all weights start at 0, or any other constant, the networks won't converge. The default initial condition for the technique has thus been to start with small random weights. In the first part of Kolen & Goel's paper, they show that if the weights aren't small, a large percentage of initial conditions lead to non-convergence. We have examined this question in more detail, by slowly varying the initial conditions to a back-propagation network, and show that, in fact, it is quite sensitive! This chaotic behavior shows up in the image

below, a 2-dimensional cut of the 7-dimensional initial weight space for the Exclusive-or network, where convergent (within 2000 iterations) and non-convergent conditions show up in black and white.

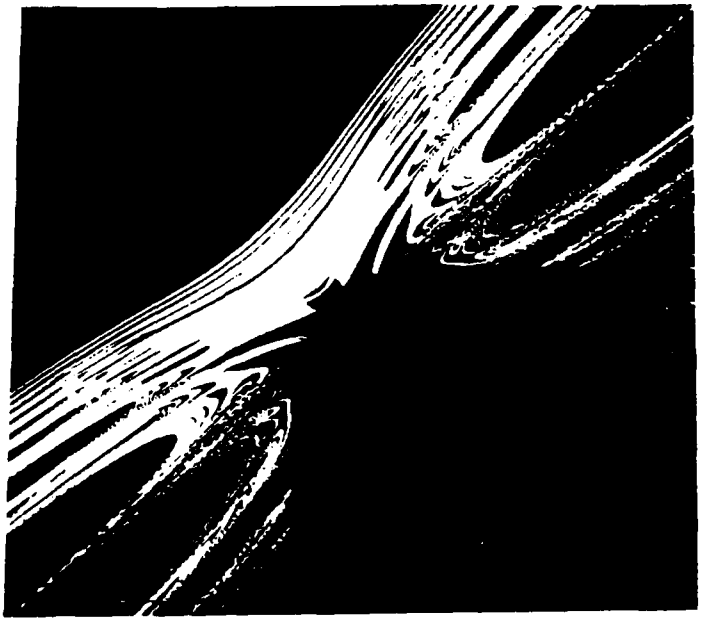

Figure 2. Strange behavior of back-propagation near boundary between convergent and non-convergent initial conditions.

Large Scale RAAM

Finally, Kolen & I have begun a larger scale study of RAAM. This work is beginning to answer two major questions about our default RAAM assumptions. The first question regards the origin of tree structures; if the world is presenting sequences (as in speech), what information processing principle (as opposed to linguistic theory) requires the construction of hierarchal representations (such as phonemes, syllables, and parse trees)? We are using direct statistical measurements (e.g. bigram frequency clustering) of the environment to get tree structure, as opposed to the manual techniques used for small examples. The second question regards the proper representations for the terminal patterns in a tree. Rather than choose random patterns for terminals which may not reflect similarity relationships, we have built a RAAM simulator extended with the idea of Miikkulainen & Dyer, to let error minimization constraints flow back into the lexicon, changing the patterns for terminals. Taken together, these two processes, of finding trees by structural clustering, and modifying lexical representations by contextual feedback, will be make our work applicable to large corpora of sequential data, and will automatically extract syntactic structures and categorize lexical items.

1. Appendices

1.1. Recursive Distributed Representations

1.2. Connectionism: Past, Present, and Future

1.3. Learning in Parallel Distributed Processing Networks

Recursive Distributed Representations

Jordan B. Pollack

Laboratory for AI Research &
Computer & Information Science Department
The Ohio State University
2036 Neil Avenue
Columbus, OH 43210
(614) 292-4890
pollack@cis.ohio-state.edu

ABSTRACT

A long-standing difficulty for connectionist modeling has been how to represent variable-sized recursive data structures, such as trees and lists, in fixed-width patterns. This paper presents a connectionist architecture which automatically develops compact distributed representations for such compositional structures, as well as efficient accessing mechanisms for them. Patterns which stand for the internal nodes of fixed-valence trees are devised through the recursive use of back-propagation on three-layer auto-associative encoder networks. The resulting representations are novel, in that they combine apparently immiscible aspects of features, pointers, and symbol structures. They form a bridge between the data structures necessary for high-level cognitive tasks and the associative, pattern recognition machinery provided by neural networks.

1. Introduction

One of the major stumbling blocks in the application of Connectionism to higher-level cognitive tasks, such as Natural Language Processing, has been the inadequacy of its representations. Both local and distributed representations have, thus far, been unsuitable for capturing the dynamically-allocated variable-sized symbolic data-structures traditionally used in AI. The limitation shows in the fact that pure connectionism has generated somewhat unsatisfying systems in this domain; for example, parsers for fixed length sentences [1-4], without embedded structures [5].[1]

Indeed, some of the recent attacks on connectionism have been aimed precisely at the question of representational adequacy. According to Minsky & Papert [10], for example, work on neural network and other learning machines was stopped by the need for AI to focus on knowledge representation in the 1970's, because of the principle that "no machine can learn to recognize X unless it possesses, at least potentially, some scheme for *representing* X (p. xiii)." Fodor and Pylyshyn's [11] arguments against connectionism are based on their belief that connectionist machines do not even have the *potential* for representing X, where X is combinatorial (syntactic) constituent structure, and hence cannot exhibit (semantic) "systematicity" of thought processes.

Agreeing thoroughly that compositional symbolic structures are important, in this paper I show a connectionist architecture which can discover compact distributed representations for them. *Recursive Auto-Associative Memory* (RAAM) uses back-propagation [12] on a non-stationary environment to devise patterns which stand for all of the internal nodes of fixed-valence trees. Further, the representations discovered are not merely connectionist implementations of classic concatenative data structures, but are in fact *new*, interesting, and potentially very useful.

The rest of this paper is organized as follows. After a background on connectionist representational schemes, the RAAM architecture is described, and several experiments presented. Finally, there is a discussion of the generative capacity of the architecture, and an analysis of the new representations and their potential applications.

---

[1] Hybrid (connectionist-symbolic) models [6-9] have the potential for more powerful representations, but do not insist on the neural plausibility constraints which create the limitations in the first place.

1.1. Background: Connectionist Representations

Normal computer programs have long used sequential data structures, such as arrays and lists as primitives. Because of the built-in notion of "address", moreover, the contents of sequences can be the addresses of other sequences; hence it is also quite simple for computer programs to represent and manipulate tree and graph structures as well. Representing lists and trees is not a trivial problem for connectionist networks, however, which do not use adjacent or randomly addressed memory cells, or permit the real-time dynamic creation of new units.

Some of the earliest work in modern connectionism made an inappropriate analogy between semantic networks and neural networks. The links in the former represented logical relations between concepts. The links in the latter represented weighted paths along which "activation energy" flowed. Needless to say, these first connectionist networks, in which each concept was mapped onto a single neuron-like unit, did not have the representational capacity of their logically powerful cousins.

Furthermore, local representational schemes do not efficiently represent sequential information. The standard approach involves converting time into space by duplicating sub-networks into a fixed set of buffers for sequential input. Both early connectionist work, such as McClelland & Rumelhart's word recognition model [13], as well as more modern efforts [4, 14] use this approach, which is not able to represent or process sequences longer than a predetermined bound. One way to overcome this length limitation is by "sliding" the input across the buffer [15, 16]. While such systems are capable of processing sequences longer than the predetermined bound, they are not really representing them.

Distributed Representations have been the focus of much research (including the work reported herein) since the circulation of Hinton's 1984 report [17] discussing the properties of representations in which "each entity is represented by a pattern of activity distributed over many computing elements, and each computed element is involved in representing many different entities."

The most obvious and natural distributed representation is a feature (or microfeature) system, traditionally used in linguistics. A good example of a connectionist model using such a representation is Kawamoto's work on lexical access [18]. However, since the entire feature system is needed to represent a single concept, attempts at

representing structures involving those concepts cannot be managed in the same system. For example, if all the features are needed to represent a NURSE, and all the features are needed to represent an ELEPHANT, then the attempt to represent a NURSE RIDING ELEPHANT may come out either as a WHITE ELEPHANT or a rather LARGE NURSE WITH FOUR LEGS.

To solve the problem of feature superposition, one might use full-size constituent buffers, such as Agent, Action, and Object [5]. In each buffer would reside a feature pattern filling these roles such as NURSE, RIDING, and ELEPHANT. Unfortunately, because of the dichotomy between the representation of a structure (by concatenation) and the representation of an element of the structure (by features), this type of system cannot represent embedded structures such as "John saw the nurse riding an elephant." A solution to the feature-buffer dichotomy problem was anticipated and sketched out by Hinton [19], and involved having a "reduced description" for NURSE RIDING ELEPHANT which would fit into the constituent buffers along with patterns for JOHN and SAW.

However, it was not immediately obvious how to develop such reduced descriptions. Instead, avant-garde connectionist representations were based on coarse-coding [17], which allows multiple semi-independent representational elements to be simultaneously present, by superposition, in a feature vector. Once multiple elements can be present, conventional groupings of the elements can be interpreted as larger structures.

For example, Touretzky has developed a coarse-coded memory system and used it in a production system [20], a primitive lisp data-structuring system called BoltzCONS [21], and a combination of the two for simple tree manipulations [22]. In his representation, the 15,625 triples of 25 symbols (A-Y) are elements to be represented, and using patterns over 2000 bits, small sets of such triples could be reliably represented. Interpreting the set of triples as pseudo-CONS cells, a limited representation of sequences and trees could be achieved.

Similarly, in their past-tense model, Rumelhart and McClelland [23] developed an *implicitly sequential representation*, where a set of well-formed overlapping triples could be interpreted as a sequence. It is instructive to view the basic idea of their representational scheme as the encoding of a sequence of tokens, $(i_1, \cdots, i_n)$ by an unordered *set* of overlapping subsequences (each of breadth k) of tokens:

$$\{(i_1, \cdots, i_k), (i_2, \cdots, i_{k+1}), \cdots, (i_{n-k+1}, \cdots, i_n)\}$$

Thus, if a coarse-coded memory can simultaneously represent a set of such subsequences, then it can also represent a longer sequence.

The limits of this type of representation are that the cost of the representation goes up exponentially with its breadth, and, for any particular breadth, there may be sequences with too much internal duplication. Sets do not count multiple occurrences of their elements. So a system, for example, which represented the spellings of words as sets of letter-*pairs* would not be able to represent the word *yoyo*, and even if the breadth were increased to three, the system would still not be able to represent words with duplicate triples such as *banana*.[2]

Although both Touretzky's and Rumelhart & McClelland's coarse-coded representations were fairly successful for their circumscribed tasks, there remain some problems:

(1) A large amount of human effort was involved in the design, compression and tuning of these representations, and it is often not clear how to translate that effort across domains.

(2) Coarse-coding requires expensive and complex access mechanisms, such as pullout networks [25] or clause-spaces [20].

(3) Coarse-coded symbol memories can only simultaneously instantiate a small number of representational elements (like triples of 25 tokens) before spurious elements are introduced[3]. Furthermore, they assume that all possible tokens need to be combined.

(4) They utilize binary codes over a large set of units (hundreds or thousands).

(5) Their mode of aggregating larger structures out of basic elements is superpositional, the cause of problems (2) and (3).

In contrast, the distributed representations devised by the RAAM architecture demonstrate better properties:

(1) Encodings are developed mechanically by an adaptive network.

---

[2] To point out this "Banana Problem" with Rumelhart & McClelland's actual representation, which was phonological rather than orthographic, Pinker and Prince [24] discovered words with enough internal duplication in the Oykangand language.

[3] Rosenfeld and Touretzky [26] provide a nice analysis of coarse-coded symbol memories.

(2) The access mechanisms are simple and deterministic.

(3) A potentially very large number of primitive elements can *selectively* combine into constituent structures. Not all triples of symbols can, or need, be represented.

(4) The representations utilize real-values over few units (tens).

(5) The aggregation mode is compositional.

2. Recursive Auto-Associative Memory

The problem under attack, then, is the representation of variable-sized symbolic sequences or trees in a numeric fixed-width form, suitable for use with association, categorization, pattern-recognition, and other neural-style processing mechanisms.

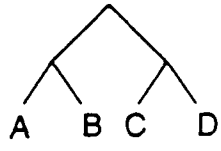

Figure 1. *Example of a binary tree.*

Consider two hypothetical mechanisms which could translate, in both directions, between symbolic trees and numeric vectors. The *Compressor* should encode small sets of fixed-width patterns into single patterns of the same size. It could be recursively applied, from the bottom up, to a fixed-valence tree with labeled terminals (leaves), resulting in a fixed-width pattern representing the entire structure. For the binary tree *((A B)(C D))*, shown in figure 1, where each of the terminals is a fixed-width pattern, this would take three steps. First A and B would be compressed into a pattern, R_1. Then C and D would be compressed into a pattern, R_2. Finally, R_1 and R_2 would be compressed into R_3.

The *Reconstructor* should decode these fixed-width patterns into facsimiles of their parts, and determine when the parts should be further decoded. It could be recursively applied, from the top down, resulting in a reconstruction of the original tree. Thus, for this example, R_3 would be decoded into R_1' and R_2'. R_1' would be decoded into A' and B', and R_2' into C' and D'.

These mechanisms are hypothetical, because it is not clear either how to physically build or computationally simulate such devices, or what the R_i patterns look like. In answer to the first question, I just assume that the mechanisms could be built out of the

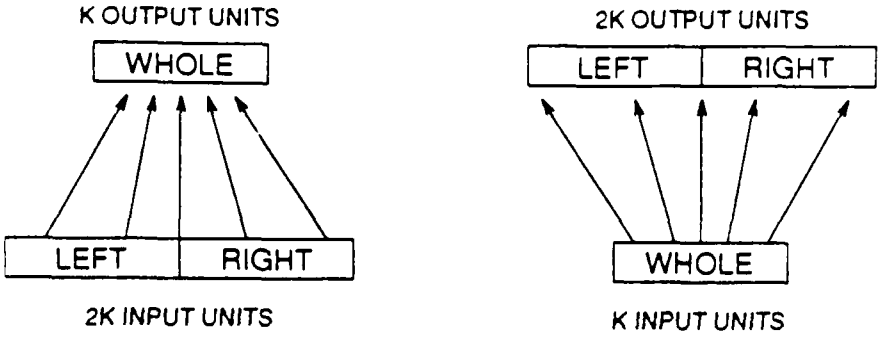

Figure 2. *Proposed feedforward networks for the Compressor and Reconstructor working with binary trees.*

standard modern connectionist substrate of layered fully-connected feed-forward networks of semi-linear units.[4] For binary trees with k-bit patterns as the leaves, the compressor could be a single-layer network with $2k$ inputs and k outputs. The reconstructor could be a single-layer network with k inputs and $2k$ outputs. Schematics for these are shown in Figure 2.

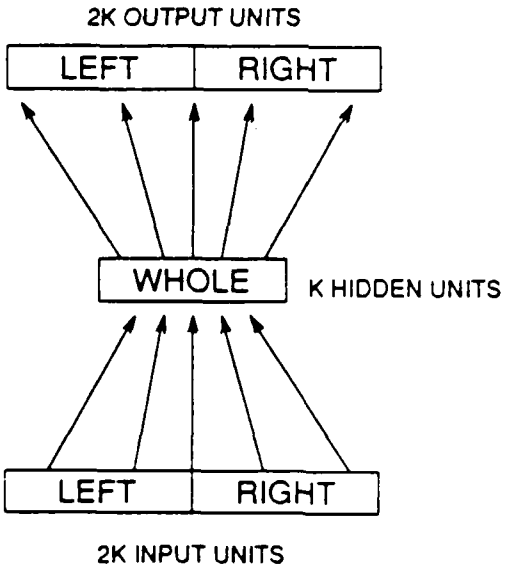

Figure 3. *Single network composed of both compressor and reconstructor.*

In answer to the second, regarding what the patterns look like, we develop the strategy of letting a connectionist network devise its own representations. Consider

[4] I also assume that the reader is, by now, familiar with this standard, as well as with the back-propagation technique for adjusting weights [12], and will not attempt a re-presentation of the mathematics. The work herein does not crucially depend on the default assumptions of semi-linearity and full-connectedness. By relying on these standard defaults, however, I hope to keep the focus on issue of representation.

simultaneously training these two mechanisms as a single $2k-k-2k$ network, as shown in Figure 3.

This looks suspiciously like a network for the Encoder Problem [27]. Back-propagation has been quite successful at this problem,[5] when used in a self-supervised auto-associative mode on a three layer network. The network is trained to reproduce a set of input patterns; i.e., the input patterns are also used as desired (or target) patterns. In learning to do so, the network develops a compressed code on the hidden units for each of the input patterns. For example, training an 8-3-8 network to reproduce the eight 1-bit-in-8 patterns usually results in a 3-bit binary code on the hidden units.

In order to find codes for trees, however, this auto-associative architecture must be used recursively (hence its name). Extending the simple example from above, if A, B, C, and D were k-bit patterns, the network could be trained to reproduce (A B), (C D), *and* ((A B)(C D)) as follows:

input pattern		*hidden pattern*		*output pattern*
$(A \quad B)$	$\rightarrow$	$R_1(t)$	$\rightarrow$	$(A'(t) \quad B'(t))$
$(C \quad D)$	$\rightarrow$	$R_2(t)$	$\rightarrow$	$(C'(t) \quad D'(t))$
$(R_1(t) \quad R_2(t))$	$\rightarrow$	$R_3(t)$	$\rightarrow$	$(R_1(t)' \quad R_2(t)')$

where t represents the time, or epoch, of training. Assuming that back-propagation converges in the limit, the sum of the squares of the differences between the desired and actual outputs would go to 0, and:

$$\begin{aligned} A' &= A \\ B' &= B \\ C' &= C \\ D' &= D \\ R_1' &= R_1 \\ R_2' &= R_2 \end{aligned}$$

Therefore, R_3, would, in fact, be a representation for the tree ((A B)(C D)), by virtue of the fact that the compressor would be a deterministic algorithm which transforms the tree to its representation, and the reconstructor a deterministic algorithm which transforms the representation back to the tree. Along the way, representations will also be devised for all subtrees, in this case, (A B) and (C D). Note that, as will be

[5] Rumelhart et al. [12] demonstrated only a 8-3-8 network, but other successful uses include a 64-16-64 network [28] and a 270-45-270 network [4]. The three numbers correspond to the number of units in the input, hidden, and output layers of a network.

demonstrated later, this strategy works on a collection of trees just as it does on a single tree.

There are a few details which form a bridge between theory and practice.

(1) The (initially random) values of the hidden units, $R_i(t)$, are used as part of the training environment. Therefore, as the weights in the network evolve, so do some of the patterns that comprise the training environment. This form of non-stationary, or "Moving Target," learning has also been explored by others [29, 30]. The stability and convergence of the network are sensitive to the learning parameters. Following the explication of Rumelhart et al. [12, p. 330], there are two such parameters: the learning rate η, which controls the the gradient descent step size, and the momentum α, which integrates the effects of previous steps. These parameters must be set low enough that the change in the hidden representations does not invalidate the decreasing error granted by the change in weights, and high enough that some change actually takes place. In the experiments described later in this paper, η was usually set to 0.1 (less for the larger experiments), and α to 0.3. As the learning curve flattens out, α is slowly increased up to 0.9, following [31].

(2) The induction relied upon is outside the mechanical framework of learning. This induction, of global success arising from only local improvements, is similar to the Bucket Brigade principle used in classifier systems [32]. Since the training strategy never reconstructs the terminals from R_1' or R_2', only the fact that they are equal, in the limit, to R_1 and R_2 allows this strategy to work.

But back-propagation cannot really run forever, and therefore, at least with use of the standard sigmoidal activation function, it is impossible to achieve the perfect encoding described above. So some practical way to decide when to stop training becomes necessary. When back-propagation is used to produce binary outputs, there is a tolerance, τ, conventionally set to 0.2, such that training can stop when every output value for every training pattern is within τ of the desired bit. For non-terminal patterns which may not be binary, however, 20% is far too permissive a tolerance. In order to successfully reconstruct A and B (to a tolerance of τ) from R_1', for example, R_1' must be *very* similar to R_1. Thus, a second tolerance, ν, is used for the real-valued non-terminals, which, for the experiments below, has been

set at 0.05.

(3) The name for this architecture, Recursive Auto-Associative Memory (RAAM), accurately reflects that the codes developed by an auto-associative memory are being further compressed. It does not reflect that there are actually two separate mechanisms which happen to be simultaneously trained. These mechanisms also require some support in the form of control and memory, but nothing beyond the ability of simple neural networks using thresholds.

In order to encode a tree from the bottom up, the compressor needs a stack on which to store temporary results (such as R_1). In order to decode a tree from the top down, the reconstructor also needs an external stack on which to store intermediate patterns. Furthermore, it needs some mechanism to perform terminal testing. In the experiments presented below, it is assumed that this terminal test is merely a threshhold test for "binary-ness", which checks that all the values of a pattern are above $1-\tau$ or below τ. Alternatively, one could train a simple classifier, or use conventional computer programs which test for membership in a set, or perform error detection and correction.

2.1. Sequential RAAM

Since sequences, such as (X Y Z), can be represented as left-branching binary trees, i.e., ((NIL X) Y) Z), an alternative version of the RAAM architecture works for developing representations and *Last-In-First-Out* access mechanisms for sequences.

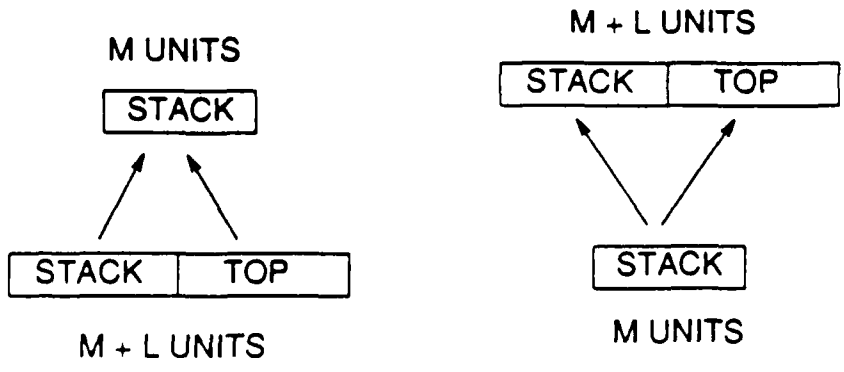

Figure 4. *Inverse sequencing mechanisms in single-layered networks. The compressor combines an m-dimensional representation for a sequence (STACK) with a new element (TOP), returning a new m-dimensional vector;*

the reconstructor decodes it back into its components.

This architecture is in fact simpler than the mechanism for trees. Compressed representations only have to be recirculated to one side, so they do have to be stored externally. There is less constraint on the size of the representations as well, and a higher dimension, M, can be assumed for the compressed patterns, than for the terminal symbols, L.

Figure 4 shows the single-layer compressor and reconstructor networks for a sequential RAAM, which, when viewed as a single network has $M+L$ input and output units, and M hidden units. An M-vector of numbers, ϵ, is assumed to stand for NIL, the empty sequence. In the experiments below, vectors of all 0.5's are chosen, which are very unlikely ever to be generated as an intermediate state. Following the earlier logic, when this network is trained with the patterns:

input pattern		*hidden pattern*		*output pattern*
$(\epsilon \quad X)$	$\rightarrow$	$R_x(t)$	$\rightarrow$	$(\epsilon'(t) \quad X'(t))$
$(R_x(t) \quad Y)$	$\rightarrow$	$R_{xy}(t)$	$\rightarrow$	$(R'_x(t) \quad Y'(t))$
$(R_{xy}(t) \quad Z)$	$\rightarrow$	$R_{xyz}(t)$	$\rightarrow$	$(R_{xy}(t)' \quad Z'(t))$

it is expected that, after back-propagation converges, R_{xyz} will be a representation for the sequence (X Y Z). Along the way, representations will also be developed for all prefixes to the sequence, in this case, (X) and (X Y).

3. Experiments with Recursive Auto-Associative Memories

3.1. Proof of Concept

To demonstrate that RAAM actually works under practical assumptions, and that it can discover compositional representations and simple access mechanisms, a small sequential RAAM is presented first.

The training set consisted of the eight possible sequences of three bits. Using a 4-3-4 network and an empty pattern of (.5 .5 .5), the representations shown in Figure 5 were developed. (The representations for all the prefixes are shown as well). The network has clearly developed into a tri-state shift-register, where the first feature corresponds to the inverse of the last bit in, the second to the inverse of the next-to-last bit, and the third to the first bit encoded.

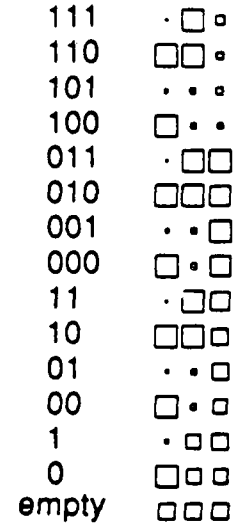

Figure 5. *Representations developed by a 4-3-4 RAAM for the complete set of bit patterns up to length 3. Each square represents a number between 0 and 1.*

A shift-register, which simply concatenates bits, is a classical means for serially constructing and accessing an obviously compositional representation. But like any finite piece of hardware built to hold a certain number of bits, it degrades rather rapidly when over-filled. The more interesting area to explore involves pattern spaces which have underlying regularities, but do not depend on representing all possible combinations of sub-patterns. It is under these conditions that an adaptive connectionist mechanism would be expected to display more desirable properties, such as content-sensitivity and graceful degradation.

3.2. Letter Sequences

Our second experiment involves learning to represent sequences of letters. Rather than trying to represent all possible sequences of letters, which would certainly give rise to another shift register, a limited subset of English words was chosen. Using an electronic spelling dictionary, those words containing only the 5 letters "B", "R", "A", "I", and "N" were selected, and then all prefixes (like "an" and "bar") were removed, resulting in the list below. Note that, in training, a representation is developed for every prefix:

AIR	ANA	ANI	BABAR	BANANA
BARBARIAN	BARN	BIBB	BIN	BRAIN
BRAN	BRIAR	INN	NAB	NIB
RABBI	RAIN	RAN	RIB	

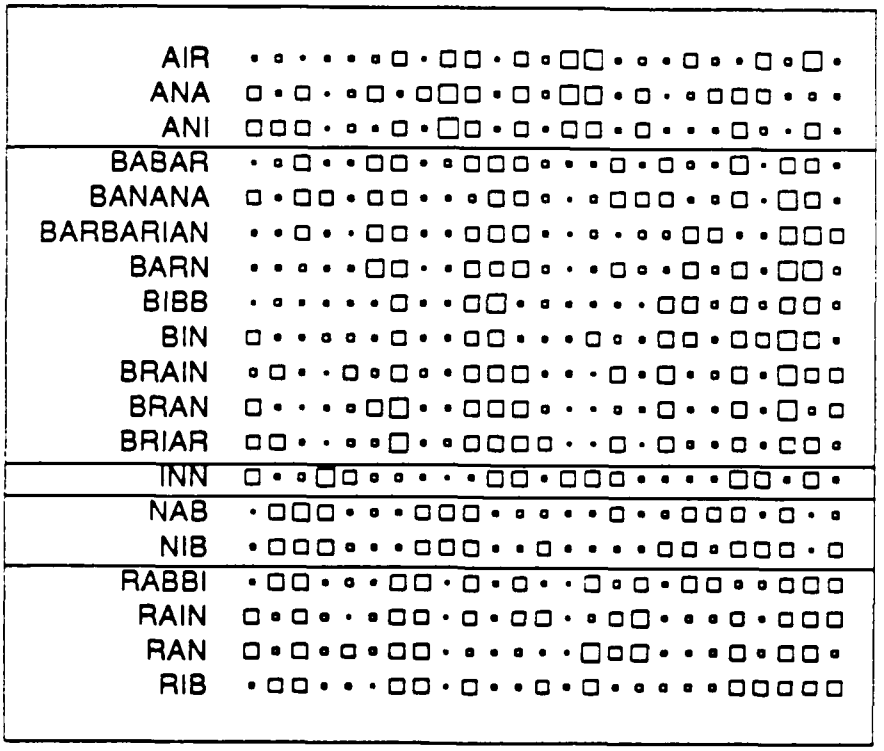

Figure 6. *Representations developed by a 30-25-30 RAAM on letter sequences.*

Each terminal was coded as a 1-in-5 bit pattern, the empty vector, again, was all 0.5's, and a 30-25-30 RAAM was used to encode these words. Note that both BANANA and BARBARIAN would be troublesome for an implicit sequential representation of breadth three. Figure 6 shows the representations for these letter sequences, and the cluster diagram in Figure 7 shows that, unlike a decaying sum representation in which information about older elements gets lost [33], this sequential representation is devoting the most resources to keeping older elements alive. And even though there are enough resources to build a 5-letter shift register, the network cannot take this easy solution path because of its need to represent the 6- and 9-letter words.

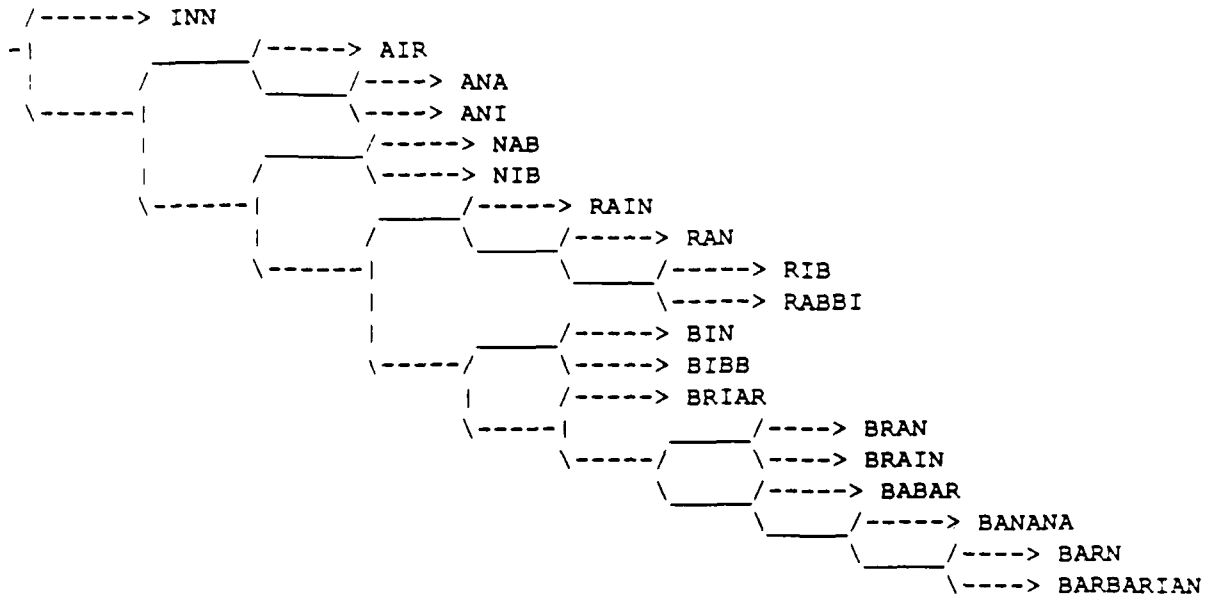

Figure 7. *Hierarchal clustering of the letter sequence representations.*

3.2.1. Learning Well-formed Syntactic Trees

The tree ((D (A N))(V (P (D N)))) might be a syntactic parse-tree for the sentence "The little boy ran up the street", given that the terminals D, A, N, V, and P stand respectively for *determiner, adjective, noun, verb,* and *preposition.* Consider a simple context-free grammar, where every rule expansion has exactly two constituents:

S -> NP VP | NP V
NP -> D AP | D N | NP PP
PP -> P NP
VP -> V NP | V PP
AP -> A AP | A N

Given a set of strings in the language defined by this grammar, it is easy to derive the bracketed binary trees which will make up a training set. With one such set of strings, a chart parser yielded the following set of trees:

(D (A (A (A N))))
((D N)(P (D N)))
(V (D N))
(P (D (A N)))
((D N) V)
((D N) (V (D (A N)))))
((D (A N)) (V (P (D N))))

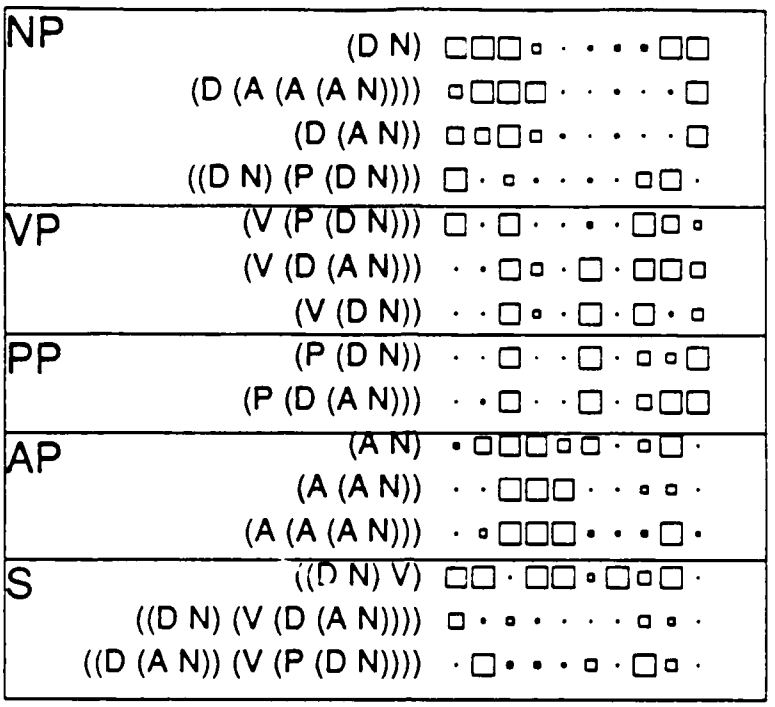

Figure 8. *Representations of all the binary trees in the training set, devised by a 20-10-20 RAAM, manually clustered by phrase-type.*

Each terminal (D A N V & P) was then represented as a 1-bit-in-5 code padded with 5 zeros. A 20-10-20 RAAM devised the representations shown in Figure 8.

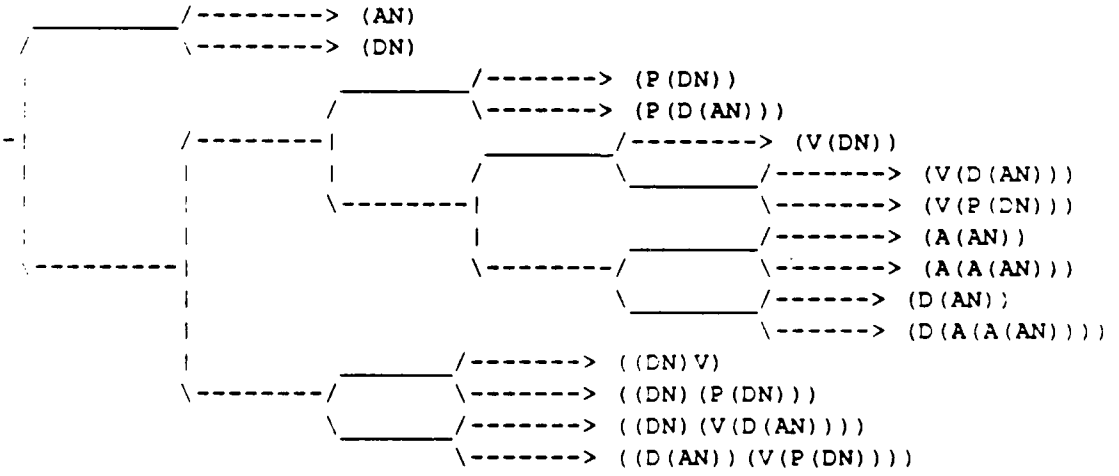

Figure 9. *Hierarchal clustering of the syntactic patterns.*

Each tree and its representation have been labeled by the phrase type in the grammar, and then sorted by type. The RAAM has clearly developed a representation with

similarity between members of the same type. For example, the third feature seems to be clearly distinguishing sentences from non-sentences, the fifth feature almost separates adjective phrases from others, while the tenth feature appears to distinguish prepositional and noun phrases from the rest.[6] Finally, a hierarchal cluster of these patterns in Figure 9 reveals that the similarity between patterns generally follows the phrase type breakup, and also reflects the depth of trees.

3.2.2. Learning to Represent Propositions.

Tree representations are common data structures, used for semantic as well as syntactic structures. This final experiment sets up some propositional representations which will be exploited later in the paper, and merely demonstrates that the architecture is capable of working on more than just binary trees.[7]

Table 1. *Collection of sentences for propositional experiment.*

1	Pat loved Mary
2	John loved Pat
3	John saw a man on the hill with a telescope
4	Mary ate spaghetti with chopsticks
5	Mary ate spaghetti with meat
6	Pat ate meat
7	Pat knew John loved Mary
8	Pat thought John knew Mary loved John
9	Pat hoped John thought Mary ate spaghetti
10	John hit the man with a long telescope
11	Pat hoped the man with a telescope saw her
12	Pat hit the man who thought Mary loved John
13	The short man who thought he saw John saw Pat

Starting with a somewhat random collection of sentences, a RAAM was used to devise compact representations for corresponding propositional forms. The sentences used for training are shown in Table 1. The terminals for this RAAM are bit patterns for the symbols which appear in these sentences minus the determiners and pronouns, plus two new symbols: *IS* is used as a subject-raiser in the representations for sentences 11 and 12, while *MOD* is used to specify adjectives in triples.

---

[6] By these metrics, of course, ((D N)(P (D N))) is being classified as an S rather than an NP. This is not surprising since, like an S, it is not being further combined.

[7] Of course, binary trees of symbols (along with a distinguished NIL element) are sufficient for arbitrary tree representations.

Table 2. *16-bit patterns for the terminal symbols*

WORD	THING 4 BITS	HUMAN 3 BITS	PREP 3 BITS	ADJ 2 BITS	VERB 4 BITS
HILL	1 0 0 0				
STREET	1 0 0 1				
TELESCOPE	1 0 1 0				
CHOPSTICKS	1 0 1 1				
MEAT	1 1 0 0				
SPAGHETTI	1 1 0 1				
MAN		1 0 0			
JOHN		1 0 1			
MARY		1 1 0			
PAT		1 1 1			
MOD			1 0 0		
WITH			1 0 1		
ON			1 1 0		
LONG				1 0	
SHORT				1 1	
IS					1 0 0 0
KNEW					1 0 0 1
HOPED					1 0 1 0
THOUGHT					1 0 1 1
LOVED					1 1 0 0
HIT					1 1 0 1
ATE					1 1 1 0
SAW					1 1 1 1

Table 3. *Ternary trees for propositional experiment.*

1 (LOVED PAT MARY)
2 (LOVED JOHN PAT)
3 ((WITH SAW TELESCOPE) JOHN (ON MAN HILL))
4 ((WITH ATE CHOPSTICKS) MARY SPAGHETTI)
5 (ATE MARY (WITH SPAGHETTI MEAT))
6 (ATE PAT MEAT)
7 (KNEW PAT (LOVED JOHN MARY))
8 (THOUGHT PAT (KNEW JOHN (LOVED MARY JOHN)))
9 (HOPED PAT (THOUGHT JOHN (ATE MARY SPAGHETTI)))
10a ((WITH HIT (MOD TELESCOPE LONG)) JOHN MAN)
10b (HIT JOHN (WITH MAN (MOD TELESCOPE LONG)))
11 (HOPED PAT (SAW (WITH MAN TELESCOPE) PAT))
12 (HIT PAT (IS MAN (THOUGHT MAN (LOVED MARY JOHN))))
13 (SAW (IS (MOD MAN SHORT) (THOUGHT MAN (SAW MAN JOHN))) PAT)

A similarity-based 16-bit binary representation was devised for the terminals, by first dividing them into 5 classes, *THING*, *HUMAN*, *PREP*, *ADJ*, and *VERB*, and then using one bit for each class along with a counter as shown in Table 2. Empty spots are all zeros. Each sentence was manually translated into a ternary tree (except sentence 10

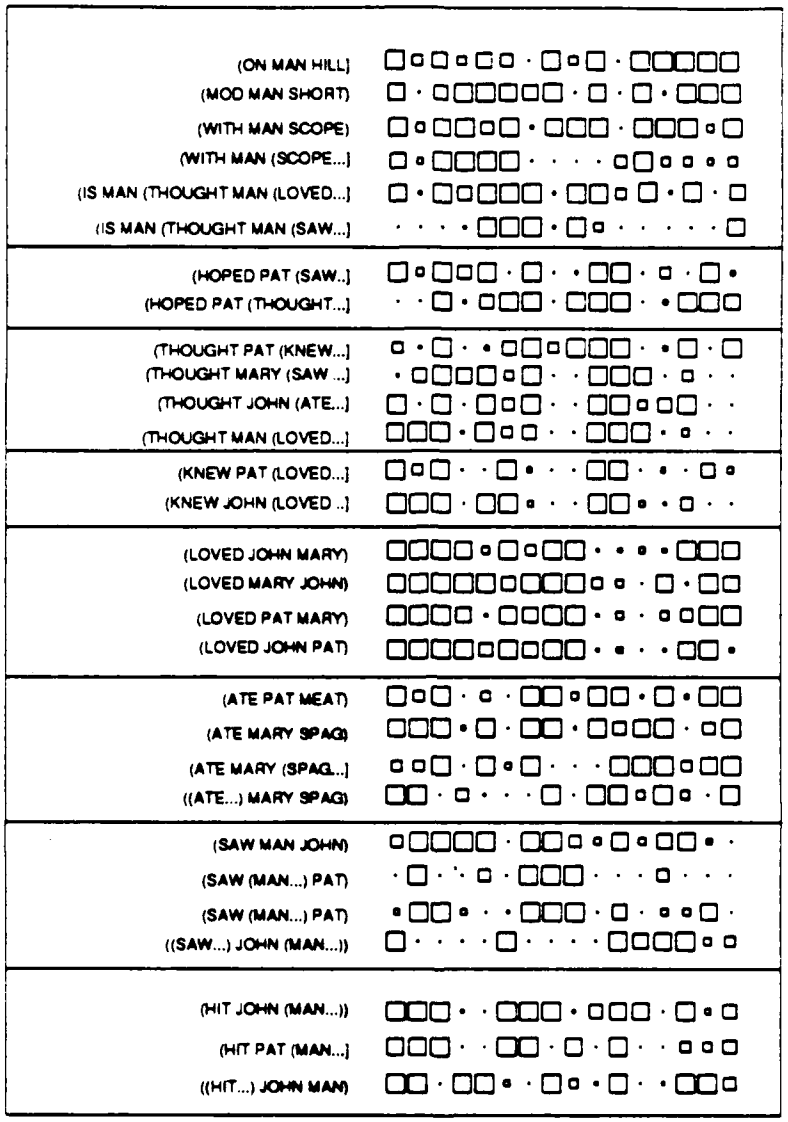

Figure 10. *Representations of the ternary semantic trees in the training set, devised by a 48-16-48 RAAM, manually clustered. The symbolic trees have been abbreviated to fit.*

which had two readings) as shown in Table 3. This representation is meant to capture the flavor of a recursive (ACTION AGENT OBJECT) case system. A 48-16-48 RAAM learned to construct representations and to recursively encode and decode these trees into their respective parts. These are again shown both pictorially (Figure 10) and clustered (Figure 11).

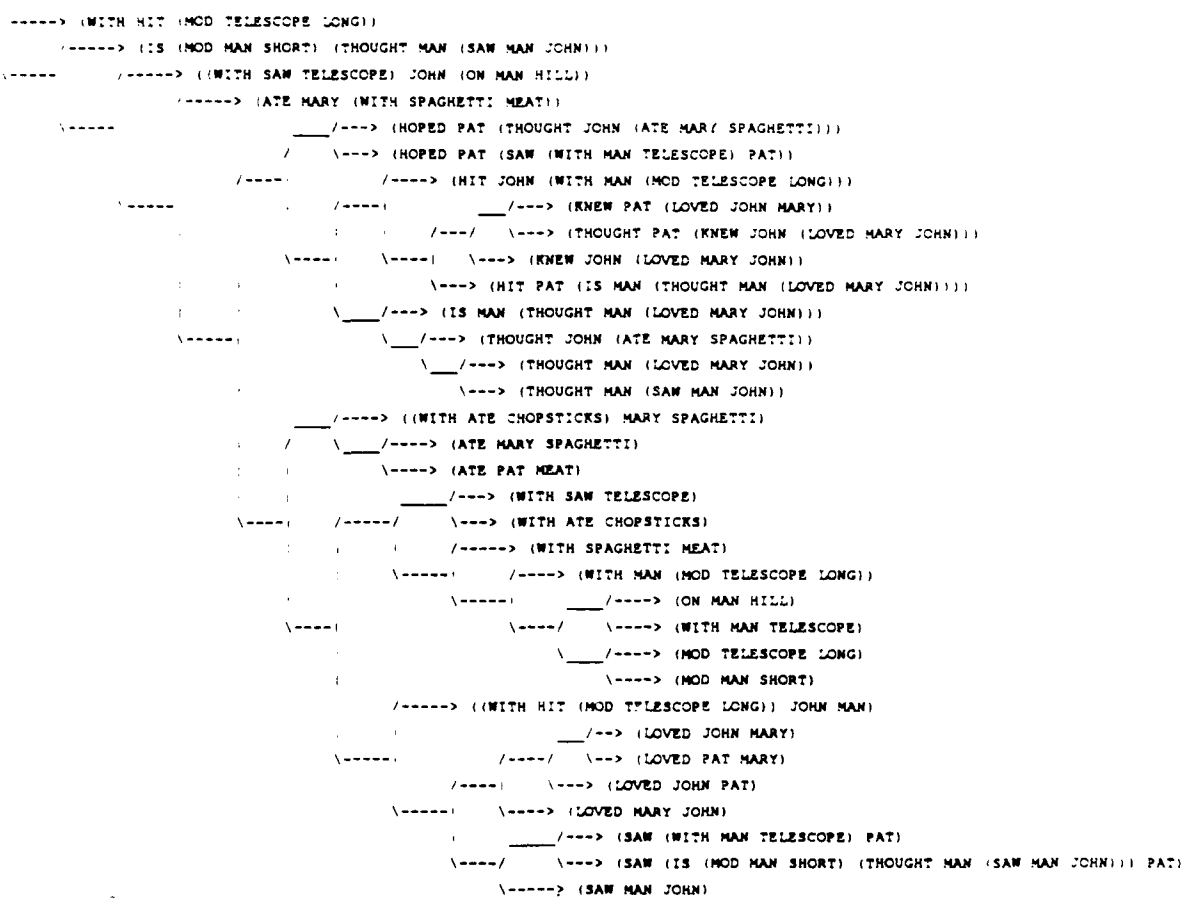

Figure 11. *Hierarchal clustering of the semantic patterns*

4. Discussion

4.1. Studies of Generalization

Perhaps the most important question about Recursive Auto-Associative Memories is whether or not they are capable of any productive forms of generalization. If it turned out that, as in the shift-register example, they were just *memorizing* the training set, finding a convenient mapping from given structures to unassigned vertices in a high-dimensional hypercube, then this work would ultimately be uninteresting. Luckily, this turns out not to be the case.

It is a straightforward matter to enumerate the set of sequences or trees that a RAAM is capable of representing, beyond the training set. Taken together, the encoder and decoder networks form a recursive well-formedness test as follows: Take two patterns for trees, encode them into a pattern for the new, higher-level, tree, and decode that

back into the patterns for the two sub-trees. If the reconstructed subtrees are within tolerance, then that tree can be considered well-formed.[8]

Using this procedure for tree RAAMs, a program can start with the set of terminals as the pool of well-formed patterns, and then exhaustively (or randomly) combine all pairs, adding new well-formed patterns to the pool. For sequential RAAMs, the pool is begun with just the pattern for the empty sequence, and a program merely attempts to compose each terminal with each pattern in the pool, adding new prefixes to the pool as they are found..

Running this generator over the network formed from the syntactic tree experiment yielded 31 well-formed trees, which are shown in Table 4. Of these, the first 12 are not really grammatical, although 8 of these seem to be based on a rule which allows two NP's to combine. There are three new instances of NP's, four new VP's, and twelve new S's. Clearly some sort of generativity, beyond memorization, is going on here, though not yet in an infinite manner. At the least, new instances of the syntactic classes are being formed by recombination of parts.

The sequential RAAM for letter sequences is quite a bit more productive. It is able to represent about 300 new sequences of letters, of which approximately one-third are wordlike, including names not in the electronic spelling dictionary like BRIAN, RINA, and BARBARA. Mostly, however, the novel sequences reflect low-order letter-transition statistics, indicating, again, that some recollective process more powerful than rote (list) memorization but less powerful than arbitrary random-access sequential storage is taking place.

There is also a tendency, especially by the 48-16-48 RAAM, to decode novel trees back to existing members of the training set. For example, the pattern encoded for (THOUGHT JOHN (KNEW PAT (LOVED MARY JOHN))) is reconstructed to (THOUGHT PAT (KNEW JOHN (LOVED MARY JOHN))), one of the original trees.

This lack of productivity is probably attributable to the problem that the input patterns are *too* similar; i.e., the Hamming distance between JOHN and PAT is only one bit. But, while this RAAM was not as productive as hoped for, it was still quite systematic,

[8] Actually, this is a bit of a simplification, since the well-formedness test does not actually guarantee that the pattern for new tree can be fully decoded. If the tolerance is kept low enough, however, the full tree will be recoverable.

Table 4. *Additional trees that can be represented by the 20-10-20 RAAM*

(D A)
(V A)
(V N)
(V V)
(((D N) (P (D N))) N)
(((D N) (P (D N))) (D (A N)))
((D N) (((D N) (P (D N))) (D (A N))))
(((D N) (P (D N))) ((D N) (P (D N))))
(((D N) (P (D N))) ((D (A N)) (P (D N))))
((D N) (((D N) (P (D N))) ((D (A N)) (P (D N)))))
(((D N) (P (D N))) (((D N) (P (D N))) (D (A N))))
(((D N) (P (D N))) (((D N) (P (D N))) ((D (A N)) (P (D N)))))

((D (A N)) (P (D N)))
((D N) (P (D (A N))))
((D (A N)) (P (D (A N))))

(V ((D N) (P (D N))))
(V ((D (A N)) (P (D N))))
(V ((D N) (P (D (A N)))))
(V ((D (A N)) (P (D (A N)))))

((D N) (V (D N)))
(((D N) (P (D N))) V)
((D N) (V ((D N) (P (D N)))))
(((D N) (P (D N))) (V (D N)))
((D N) (V ((D (A N)) (P (D N)))))
((D N) (V ((D N) (P (D (A N))))))
(((D N) (P (D N))) (V (D (A N))))
((D N) (V ((D (A N)) (P (D (A N))))))
(((D N) (P (D N))) (V ((D N) (P (D N)))))
(((D N) (P (D N))) (V ((D N) (P (D (A N))))))
(((D N) (P (D N))) (V ((D (A N)) (P (D N)))))
(((D N) (P (D N))) (V ((D (A N)) (P (D (A N))))))

according to Fodor & Pylyshyn's [11, p. 39] own definition:

> *What does it mean to say that thought is systematic? Well, just as you don't find people who can understand the sentence 'John loves the girl' but not the sentence 'the girl loves John,' so too you don't find people who can think the thought that John loves the girl but can't think the think the thought that the girl loves John.*

All 16 cases of (LOVED X Y), with X and Y chosen from the set {JOHN, MARY, PAT, MAN} were able to be reliably represented, even though only four of them were in the training set.

4.1.1. Improving Generalization Capacity

The productive capacity of these systems is not yet what it should be. There ought to be some way to acquire, at least theoretically, the ability to represent infinite numbers of similar structures in such recursive distributed representations.

Given that the simplest formulation (i.e., a 3-layer fully-connected semi-linear network) using rather arbitrary training sets has shown some limited capacity in the form of a small number of new useful representations composed out of existing constituents, it seems likely that (1) better training environments and (2) different mathematical assumptions will be needed.

First, the similarity and difference relationships between terminal patterns affects the productivity of a RAAM. In the case of the semantic triples, the fact that terminals in the same class, like JOHN and MARY, were assigned very similar patterns, lead both to their ability to be used systematically, and to the problem that single-bit errors in reconstruction were damaging. On the other hand, one would expect fully random patterns to not generalize very well either. This brings up the question of how to design compressible representations. It seems very likely that the same sort of representations devised by a RAAM for the non-terminal patterns would lead to the best possible compression and generalization properties if adopted for terminals.

Secondly, to achieve truly infinite representational capacity in fixed-width patterns, it will be necessary, at least theoretically, to consider the underlying mathematical basis for connectionist networks, freed from the default implementational assumptions of back-propagation, i.e., floating-point calculations of linear combinations and sigmoids. On the one hand, it must be considered whether or not to use real numbers at all since they are seem biologically and computationally problematic. An unbounded number of bits can be trivially compressed into a real number, leading to unbounded storage and communication costs. A simulated connectionist system using real numbers might be able to use these bits, (i.e. in very precise output values) without properly paying for them. By using only a binary code, a system must be able to to exploit the redundancy (i.e. sparseness or regularity) in the environment. On the other hand, it is certainly reasonable, however unbiological, to assume rational numbers for a competence theory. The question to answer is whether there is a similarity-preserving mapping from complex structured representations to high-dimensional spatial representations.

4.2. Analysis of the Representations

I do not yet have a prescription for engineering recursive distributed representations, but have a few insights into how they work. Top-down and bottom-up constraints work together to forge the representations. The bottom-up constraint is that each pattern is completely determined by its constituents and the knowledge eventually fixed in the network weights: *Trees with similar constituents must be similar.* The top-down constraint is that redundant information must be compressed out of similar structures (such as two NP's which both can combine with the same VP): *The possible siblings of a pattern must be similar.* Working against this drive towards similarity is the system-wide goal of minimizing error, which serves to "constrain apart" the patterns for different trees in the environment. The result of these pressures is that these representations consist of at least two types of features: *Categorical* features, such as those identified earlier as being able to separate classes, and *distinctive* features, which vary across, and discriminate between, the members of each class.

The categorical features developed by the syntactic tree experiment become clear in examination of the of a small classifier. The patterns for each tree in the training set were used as input to a 10-input 5-output network which was trained to discriminate the classes NP, VP, PP, AP, and S.

Table 5. *Weights of single-layer classifier network rounded to integers.*

	NP	VP	PP	AP	S	Strength
Bias	-2	-8	-3	-4	6	
1	8	0	-2	-5	-4	19
2	2	-8	-3	-1	5	19
3	0	7	-2	3	-9	21
4	-1	2	-5	5	-1	14
5	-5	-6	-1	3	-1	16
6	-3	3	4	0	-4	14
7	-2	-1	0	-5	3	11
8	-10	10	-4	-5	2	31
9	3	0	2	2	-4	11
10	4	-9	7	-6	-3	29

Table 5 shows all the weights in this network, rounded to integers. The columns correspond to the categories, and the rows correspond to the features. The bias inputs to the category units are also shown as the first row, as are the sums of the absolute values of the weights in each row. Looking at the column labeled NP, for example, it is clear that the first, ninth, and tenth features strongly code for NP, while the eighth and fifth

features code against NP. Looking at the column labeled VP, the third and eight features code for it, and the second and tenth against.

The "strength" of each row indicates how categorical or distinctive a feature is. The tenth feature, for example, strongly codes for NP and PP and against VP, AP, and S. The features which do not connect strongly everywhere, like the seventh and ninth, are used for discriminations within the categories. With regard to the binary-versus-real question raised earlier, it seems that RAAM may build a hybrid code. Strong binary distinctions are used for categorical judgements, while weaker analog distinctions are used for discriminating (and labeling) members within the categories.

4.2.1. Geometric Interpretation

An alternative means of understanding these representations may come from geometry. The terminal patterns are vertices of a *k*-dimensional hypercube which contains all of the non-terminal patterns.

For binary trees, a RAAM is finding a consistent invertible mapping which works the same way on composable pairs of vertices, as it does on the internal points that are also composed. To view an image of this, a 6-3-6 RAAM was trained on the two trees ((A B)(C D)) and ((A C)(B D)), with A = (0 0 0), B = (1 0 0), C = (0 1 0), and D = (1 1 0); i.e. with A, B, C, and D the four points on the "floor" of a 3-D cube.

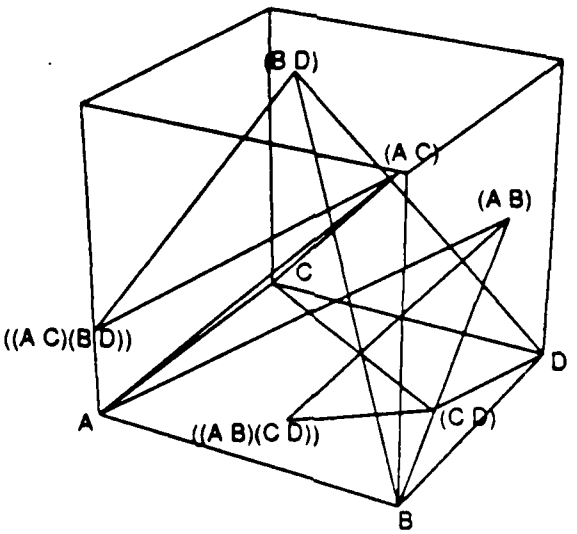

Figure 12. *Perspective diagram for the 3-dimensional codes developed for the trees ((A B)(C D)) and ((A C)(B D)).*

Figure 12 shows a perspective plot of the 3-dimensional hypercube for the codes developed for these two trees. If one stares long enough, taking each pair of composable points in one's mental left and right hands, one can see triangles falling forward as they reduce in scale.

Saund [34] has investigated (non-recursive) auto-association as a method of *dimensionality reduction*, and asserted that, in order to work, the map must be constrained to form a small dimensional parametric surface in the larger dimensional space. Consider just a 2-1-2 auto-associator. It is really an invertible mapping from certain points on the unit square to points on the unit line. In order to work, the network might develop a parametric 1-dimensional curve in 2-space, perhaps a set of connected splines. As more and more points need to be encoded, this parametric curve must get "curvier" to cover them. In the limit, especially if there are any dense "patches" of 2-space which need to be covered, it can no longer be a 1-dimensional curve, but must become a space-filling curve with a fractal dimension [35]. The notions of associative and reconstructive memories with fractal dimensions are further discussed elsewhere [36].

4.3. Applications

4.3.1. Associative Inference

Since RAAM can devise representations of trees as numeric vectors which then can be attacked with the fixed-width techniques of neural networks, this work might lead to very fast inference and structural transformation engines. The question, of course, is whether the patterns for trees can be operated on, in a systematic fashion, without being decoded first. Below is a very simple demonstration of this possibility.

Since the RAAM for the propositional triples was able to represent all 16 cases of (LOVED X Y), it should be possible to build an associative network which could perform the simple implication: "If (LOVED X Y) then (LOVED Y X)". This would be a trivial shifting task if performed on an explicit concatenative representation. However, since the (48 bit) triples are compressed into 16-dimensional pattern vectors, it is not quite as simple a job.

The task is to find an associator which can transform the compressed representation for each antecedent (e.g. (LOVED MARY JOHN)) into the compressed representation

for its consequent (e.g. (LOVED JOHN MARY)). Using back-propagation, a 16-8-16 feed-forward network was trained on 12 of the 16 pairs of patterns (to within 5% tolerance) and was then able to successfully transform the remaining 4 pairs.

What about a system which would need to follow long chains of such implications? There has recently been some work showing that under certain conditions, feed-forward networks with hidden layers can compute arbitrary non-linear mappings [37-39]. Therefore, I anticipate that the sequential application of associative inference will be able to be compiled, at least by slow training, into fast networks of few layers.

Consider homogenous coordinate transformations (in computer graphics), where the linear nature of the primitive operations (scaling, rotation, and translation) allows any sequence of them to be "compiled" into a single matrix multiplication. The field of AI has not, to date, produced any compiling methods which can rival this speedup, because most interesting AI problems are nonlinear and most interesting AI representations are not numeric. The point is that given suitable representations, efficient non-linear mapping engines could generate significant speed improvements for inferential processing.

4.3.2. Massively Parallel Parsing, Revisited

I introduced this paper by noting that natural language processing posed some problems for connectionism, precisely because of the representational adequacy problem. One cannot build either a parser or a generator without first having good "internal" representations. RAAMs can devise these compositional representations, as shown by the experiment on semantic triples, which can then be used as the target patterns for recurrent networks which accept sequences of words as input.

A feasibility study of this concept has been performed as well, using a sequential cascaded network [40], a higher-order network with a more restricted topology than Sigma-Pi [41]. Basically, a cascaded network consists of two subnetworks: The *function network* is an ordinary feed-forward network, but its weights are dynamically computed by the purely linear *context network*, whose outputs determine each weight of the function net. In a sequential cascaded network, the outputs of the function network are directly fed back to the inputs of the context network. This network is trained with presentations of initial context, input sequences, and desired final state.

Table 6. *10-bit input patterns for connectionist parser.*

WORD	CLASS	IDENTITY
JOHN	1 0 0 0 0	1 1 0 0 0
MAN	1 0 0 0 0	0 1 0 0 0
PAT	1 0 0 0 0	1 1 1 0 0
MARY	1 0 0 0 0	1 0 1 0 0
HE/HER	1 0 0 0 0	0 1 0 1 0
TELESCOPE	0 1 0 0 0	0 0 1 0 1
SPAGHETTI	0 1 0 0 0	1 0 0 1 0
CHOPSTICKS	0 1 0 0 0	0 0 1 1 0
HILL	0 1 0 0 0	0 1 0 0 0
MEAT	0 1 0 0 0	1 0 0 0 1
ON	0 0 1 0 0	1 0 0 0 0
WITH	0 0 1 0 0	0 1 0 0 0
WHO	0 0 1 0 0	0 0 1 0 0
BY	0 0 1 0 0	0 0 0 1 0
ATE	0 0 0 1 0	0 0 1 0 0
HIT	0 0 0 1 0	0 0 0 1 0
SAW	0 0 0 1 0	0 0 0 0 1
LOVED	0 0 0 1 0	0 0 0 1 1
HOPED	0 0 0 1 0	0 1 1 0 0
THOUGHT	0 0 0 1 0	0 1 0 1 0
KNEW	0 0 0 1 0	0 1 0 0 1
LONG	0 0 0 0 1	0 0 0 1 0
SHORT	0 0 0 0 1	0 0 0 0 1

A new 10-bit similarity-based encoding was created for the words appearing in the sentences, making HE and HER identical. The first 5 bits define the class, and the second 5 bits distinguish the members. The patterns are displayed in Table 6. A sequential cascaded network consisting of a 10-10-16 function network and a 16-286 context network was trained using sequences of these bit patterns corresponding to the sentences in Table 1. The initial context vectors were all zeroes, and the desired final states were the compressed 16-dimensional representations devised by the 48-16-48 RAAM for the trees in Table 3 (not including 10b).

This system is the closest thing yet to a barely adequate connectionist system for processing language: Given a variable-length sequence of words, the network returns, in linear time, a 16-dimensional vector, which can be decoded into a "meaning" by a RAAM, and can perhaps be operated upon by associative inference engines.

On the one hand, this system has extreme deficiencies if it is evaluated as a cognitive model. It can only produce a single tree for a sentence, and only handles a very small corpus of sentences. The simplifying assumption, that internal representations can first be devised and then used as target patterns, is questionable. On the other, the system has

some very interesting aspects. Besides the fact that it runs in linear time and outputs a compositional representation for the sentences, it automatically performs prepositional phrase attachment (i.e., correctly parses the "MARY ATE SPAGHETTI WITH MEAT/CHOPSTICKS" examples) and pronoun resolution (i.e., automatically replaces HE or HER with the proper filler). Finally, it is the first connectionist parser which can deal with embedded structures without resorting to external symbolic computational power.

4.4. Further Work

There is a great deal of research still to be conducted in this area, besides the conversion of the small feasibility studies into both falsifiable cognitive models and reliably engineered artifacts. Immediate concerns include:

- Understanding the convergence and stability properties of the "moving target" learning strategy; both empirical and analytical studies are called for. Similarly, the relationship between the termination condition (using τ an ν) and the depth capacity of RAAM needs to be better understood..
- Developing a complete understanding of the representations and mechanisms which are developed. A good outcome would be a general representational scheme which could be analytically derived for a particular representational task without relying on slow, gradient-descent learning.

5. Conclusion

Here is a conundrum for theories of human and machine learning: *Which came first*, the mental procedure or the mental representation? Minsky and Papert claimed that the representational egg must come before the procedural chicken, while Fodor and Pylyshyn claimed to intimately know the egg and, by extension, the exclusive class of fertile chickens. The flip side, of course, is that this perfect egg may only be layable by an impossible chicken: A formal representational theory, specified without consideration of its own genesis, may not be learnable by any mechanism in principle.

This work points to biologically certified way out of the dilemma: **Co-Evolution.** The representations and their associated procedures develop slowly, responding to each other's constraints through a changing environment. The constraint that the

representations fit into fixed-width patterns interacts with the constraint that the patterns must compose in certain well-formed ways, giving rise to fixed-width patterns which capture structural similarity in spatial distance.

The RAAM architecture has been inspired by two powerful ideas. The first is due to Hinton [42], who showed that, when properly constrained, a connectionist network can develop semantically interpretable representations on its hidden units. The second is an old idea, that given a sufficiently powerful form of learning, a machine can learn to efficiently perform a task by example, rather than by design. Taken together, these ideas suggest that, given a task, specified by example, which *requires* embedded representations, a network might be able to develop these representations itself.

It turns out that there is no *single* task which requires such representations. There have to be at least two tasks; one to construct the representations, and another to access them. On address-based machines, these tasks, such as string concatenation and array indexing, are so computationally primitive and natural that they fall far below notice. They are not natural to neural networks and thus need to be examined anew. Here, the resulting task-specific mechanisms, the compressor and reconstructor, together form a *reconstructive* memory system, in which only traces of the actual memory contents are stored, and reliable facsimiles are created with the use of domain knowledge.

The systematic patterns developed by RAAM are a very new kind of representation, a **recursive, distributed representation**, which seems to instantiate Hinton's notion of the "reduced description" mentioned earlier [19]. They combine apparently immiscible aspects of well-understood representations: They act both like feature vectors with their fixed width and simple measures of similarity, and like pointers, so that, with simple efficient procedures their contents can be "fetched." Even further, they act like compositional symbol structures: Simple associative procedures, such as the reconstructor, pattern classifiers, and pattern transformers, are clearly sensitive to their internal structure.

However, unlike feature vectors, these representations recursively combine into constituent structures, according to statistically inferred well-formedness constraints. Unlike pointers (or symbols like G0007), they contain information suitable for similarity measurements and, thus, nearest-neighbor judgements. And, unlike symbol structures, they can be easily compared, and do not have to be taken apart in order to be worked on. Recursive distributed representations may thus lead to a reintegration of the syntax and

semantics at a very low level.

Currently, symbolic systems use information-free "atoms" which physically combine (through bit or pointer concatenation) in a completely unrestricted fashion. Thus, for any domain, a syntax is required to restrict those "molecules" *after the fact*, to the set of semantically interpretable ones. With further work, recursive distributed representations might undergo a metamorphism into symbols which contain their own meanings and physically combine *only* in a systematic fashion. After all, real atoms and molecules do so all the time.

Acknowledgements

This work has been partially supported by the the State Legislatures of New Mexico and Ohio and by the Office of Naval Research, under grant N00014-89-J-1200. Thanks to Tony Plate, who implemented back-propagation in C, and to Yoshiro Miyata for a copy of his hierarchal cluster program. Comments from B. Chandrasekaran, G. Hinton, J. McClelland, D. Touretzky, T. VanGelder, and many, many others helped to improve this presentation.

References

1. G. W. Cottrell, Connectionist Parsing, *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA, 1985.
2. M. Fanty, Context-free parsing in Connectionist Networks, TR174, University of Rochester, Computer Science Department, Rochester, N.Y., 1985.
3. B. Selman, Rule-Based Processing in a Connectionist System for Natural Language Understanding, CSRI-168, University of Toronto, Computer Systems Research Institute, Toronto, Canada, 1985.
4. S. J. Hanson and J. Kegl, PARSNIP: A connectionist network that learns natural language grammar from exposure to natural language sentences, *Proceedings of the Ninth Conference of the Cognitive Science Society*, Seattle, 1987, 106-119.
5. J. McClelland and A. Kawamoto, Mechanisms of Sentence Processing: Assigning Roles to Constituents, in *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*, vol. 2, J. L. McClelland, D. E. Rumelhart and the PDP research Group (ed.), MIT Press, Cambridge, 1986.
6. J. B. Pollack and D. L. Waltz, Natural Language Processing Using Spreading Activation and Lateral Inhibition, *Proceedings of the Fourth Annual Cognitive Science Conference*, Ann Arbor, MI, 1982, 50-53.
7. D. L. Waltz and J. B. Pollack, Massively Parallel Parsing: A strongly interactive model of Natural Language Interpretation, *Cognitive Science* 9, 1 (1985), 51-74.
8. W. G. Lehnert, Case-based problem-solving with a large knowledge base of learned cases, *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, 1987, 301-306.
9. G. Berg, A Parallel Natural Language Processing Architecture with Distributed Control, *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, 1987, 487-495.
10. M. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1988.

11. J. Fodor and A. Pylyshyn, Connectionism and Cognitive Architecture: A Critical Analysis, *Cognition 28*, (1988), 3-71.

12. D. E. Rumelhart, G. Hinton and R. Williams, Learning Internal Representations through Error Propagation, in *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*, vol. 1, D. E. Rumelhart, J. L. McClelland and the PDP research Group (ed.), MIT Press, Cambridge, 1986, 25-40.

13. J. L. McClelland and D. E. Rumelhart, An interactive activation model of the effect of context in perception: Part 1. An account of basic findings, *Psychology Review 88*, (1981), 375-407.

14. R. Allen, Several Studies on Natural Language and Back Propagation, *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*, San Diego, 1987, II-335-342.

15. T. J. Sejnowski and C. R. Rosenberg, Parallel Networks that Learn to Pronounce English Text, *Complex Systems 1*, (1987), 145-168.

16. E. Charniak and E. Santos, A context-free connectionist parser which is not connectionist, but then it is not really context-free either., in *Advances in Connectionist & Neural Computation Theory*, J. Barnden and J. Pollack (ed.), Ablex, Norwood, NJ, 1989.

17. G. E. Hinton, Distributed Representations, CMU-CS-84-157, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA, 1984.

18. A. H. Kawamoto, Dynamic Processes in the (Re)Solution of Lexical Ambiguity, Doctoral Dissertation, Department of Psychology, Brown University, Providence, 1985.

19. G. Hinton, Representing Part-Whole hierarchies in connectionist networks, *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Montreal, 1988, 48-54.

20. D. S. Touretzky and G. E. Hinton, Symbols among the neurons: details of a connectionist inference architecture, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985.

21. D. S. Touretzky, BoltzCONS: Reconciling connectionism with the recursive nature of stacks and trees, *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, Amherst, MA, 1986, 522-530.

22. D. S. Touretzky, Representing and transforming recursive objects in a neural network, or "trees do grow on Boltzmann machines", *Proceedings of the 1986 Institute of Electrical and Electronics Engineers International Conference on Systems, Man, and Cybernetics*, Atlanta, GA, 1986.

23. D. E. Rumelhart and J. L. McClelland, On Learning the Past Tenses of English Verbs, in *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*, vol. 2, J. L. McClelland, D. E. Rumelhart and the PDP research Group (ed.), MIT Press, Cambridge, 1986, 216-271.

24. S. Pinker and A. Prince, On Language and Connectionism: Analysis of a parallel distributed processing model of language inquisition., *Cognition 28*, (1988), 73-193.

25. M. Mozer, Inductive information retrieval using parallel distributed computation, Technical Report, Institute for Cognitive Science, UCSD, La Jolla, 1984.

26. R. Rosenfeld and D. Touretzky, Four capacity models for coarse-coded symbol memories, *Complex Systems 2*, (1988), 463-484.

27. D. H. Ackley, G. E. Hinton and T. J. Sejnowski, A learning algorithm for Boltzmann Machines, *Cognitive Science 9*, (1985), 147-169.

28. G. Cottrell, P. Munro and D. Zipser, Learning Internal Representations from Gray-Scales Images: An Example of Extensional Programming., *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Seattle, 1987, 461-473.

29. J. L. Elman, Finding Structure in Time, Report 8801, Center for Research in Language, UCSD, San Diego, 1988.

30. R. Miikkulainen and D. G. Dyer, Forming Global Representations with Back Propagation, *Proceedings of the Institute of Electrical and Electronics Engineers Second Annual International*

Conference on Neural Networks, San Diego, 1988.

31. D. C. Plaut, S. Nowlan and G. E. Hinton, Experiments on learning by back-propagation, CMU-CS-86-126, Computer Science Dept., Carnegie Mellon University, Pittsburgh, 1986.
32. J. H. Holland, K. J. Holyoak, R. E. Nisbett and P. R. Thagard, *Induction: Processes of Inference, Learning, and Discovery*, MIT Press, Cambridge, 1986.
33. M. I. Jordan, Serial Order: A Parallel Distributed Processing Approach, ICS report 8608, Institute for Cognitive Science, UCSD, La Jolla, 1986.
34. E. Saund, Dimensionality Reduction and Constraint in Later Vision, *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, 1987, 908-915.
35. B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, San Francisco, 1982.
36. J. B. Pollack, Implications of Recursive Distributed Representations, in *Advances in Neural Information Processing Systems*, D. Touretzky (ed.), Morgan Kaufman, Los Gatos, CA, 1989.
37. K. Hornik, M. Stinchcombe and H. White, Multi-layer Feedforward Networks are Universal Approximators, *Neural Networks*, To Appear.
38. R. P. Lippman, An introduction to computing with neural networks, *Institute of Electrical and Electronics Engineers ASSP Magazine April*, (1987), 4-22.
39. A. S. Lapedes and R. M. Farber, How Neural Nets Work, LAUR-88-418, Los Alamos, 1988.
40. J. B. Pollack, Cascaded Back Propagation on Dynamic Connectionist Networks, *Proceedings of the Ninth Conference of the Cognitive Science Society*, Seattle, 1987, 391-404.
41. R. Williams, The Logic of Activation Functions, in *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*, vol. 1, D. E. Rumelhart, J. L. McClelland and the PDP research Group (ed.), MIT Press, Cambridge, 1986, 423-443.
42. G. E. Hinton, Learning Distributed Representations of Concepts, *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, 1986, 1-12.

Artificial Intelligence Review (1989) 3, 3–20

Connectionism: past, present, and future

J.B. Pollack
Department of Computer and Information Science, The Ohio State University.

Abstract Research efforts to study computation and cognitive modeling on neurally-inspired mechanisms have come to be called Connectionism. Rather than being brand new, it is actually the rebirth of a research programme which thrived from the 40s through the 60s and then was severely retrenched in the 70s. Connectionism is often posed as a paradigmatic competitor to the Symbolic Processing tradition of Artificial Intelligence (Dreyfus & Dreyfus, 1988), and, indeed, the counterpoint in the timing of their intellectual and commercial fortunes may lead one to believe that research in cognition is merely a zero-sum game. This paper surveys the history of the field, often in relation to AI, discusses its current successes and failures, and makes some predictions for where it might lead in the future.

1. Early endeavours: high hopes and hubris

Before the explosion of symbolic artifical intelligence, there were many researchers working on mathematical models of intelligence inspired by what was known about the architecture of the brain. Under an assumption that the mind arises out of the brain, a reasonable research path to the artificial mind was to simulate the brain to see what kind of mind could be created. At the basis of this programme was the assumption that neurons were the information processing primitives of the brain, and that reasonable models of neurons connected into networks would suffice.

McCulloch & Pitts

The opening shot in neural network research was the 1943 paper by Warren S. McCulloch and Walter Pitts. In 'A logical calculus of ideas immanent in nervous activity' they proved that any logical expression could be 'implemented' by an appropriate net of simplified neurons.

They assumed that each 'neuron' was binary and had a finite threshold, that each 'synapse' was either excitatory or inhibitory and caused a finite delay (of one cycle), and that networks could be constructed with multiple synapses between any pair of nodes. In order to show that any logical expression is computable, all that is

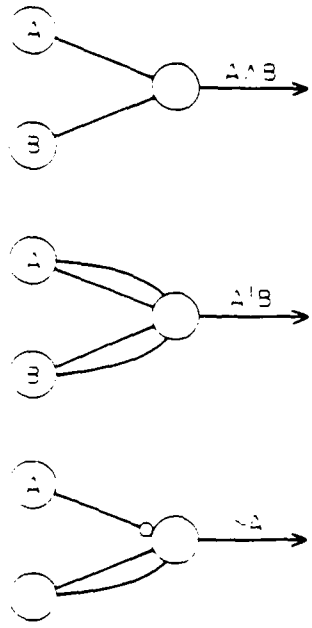

Fig. 1 Logical primitives AND, OR and NOT implemented with McCulloch & Pitts neurons. A neuron 'fires' if it has at least two activating synapses (arrow links) and no inhibiting inputs (circle links).

necessary is to build the functions AND, OR and NOT. Fig. 1 shows the simple networks which accomplish these functions. And in order to build 'larger' functions, one need only glue these primitives together. For example, Fig. 2 shows a two-layer network which computes the exclusive-or function. Continuing in this vein, one could construct a computer by building up the functional parts, e.g. memories, Arithmetic Logic Units (ALUs), from smaller pieces, which is exactly how computers are built.

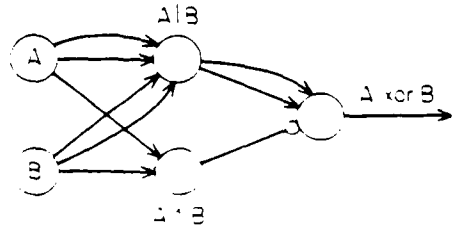

Fig. 2 A two-layer McCulloch & Pitts network which computes exclusive-or as the function $A \cdot \bar{B} + \bar{A} \cdot B$.

McCulloch and Pitts proved several theorems about equivalences of different processing assumptions, both for simple nets and for nets with feedback cycles, using a somewhat arcane syntax of temporal propositions. Since learning was not under consideration, memory, for them, was based on 'activity [which] may be set up in a circuit and continue reverberating around it for an indefinite period of time'.[1] They concluded with a discussion of Turing computability, which, for their nets, required an *external* tape.

Hebb

There was very little psychology in the science of neural nets, and very few neural considerations in the mainly stimulus–response psychology of the day. In *The Organization of Behavior*, Donald O. Hebb set out to rectify this situation, by developing a physiologically-motivated theory of psychology.

Rejecting reflexes, Hebb put forth and defended the notion of an *autonomous central process*, which intervenes between sensory input and motor output. Of his own work he said:

> 'The theory is evidently a form of *connectionism*, one of the switchboard variety, though it does not deal in direct connections between afferent and efferent pathways: not an "S–R" psychology if R means a muscular response. The connections serve rather to establish autonomous central activities, which then are the basis of further learning.'[2]

Of an incredibly rich work, Hebb is generally credited with two notions that continue to hold influence on research today. The first is that memory is stored in connections and that learning takes place by synaptic modification:

> 'Let us assume then that the persistence or repetition of a reverberatory activity (or 'trace') tends to induce lasting cellular changes that add to its stability. The assumption can be precisely stated as follows: *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*'[3]

And the second is that neurons do not work alone, but may, through learning, become organized into larger configurations, or 'cell-assemblies', which could thus perform more complex information processing.

Ashby

In *Design for a Brain*, W. Ross Ashby laid out a methodology for studying adaptive systems, a class of machines to which, he asserted, the brain belongs. He set out an ambitious program:

> [...] we must suppose (and the author accepts) that a real solution of our problem will enable an artificial system to be made that will be able, like the living brain, to develop adaptation in its behaviour. Thus the work, if successful, will contain (at least by implication) a specification for building an artificial brain that will be similarly self-co-ordinating.[4]

While the work was not 'successful' in these terms, Ashby laid the groundwork for research that is flourishing today. His methodology for studying dynamical systems as fields of variables over time is echoed today in the connectionist studies which involve time evolution of dynamical systems (Hopfield, 1982; Smolensky, 1986) and his notion of building intelligent machines out of homeostatic elements can be seen as precursor to Klopf's (1982) work on heterostatic elements.

Rosenblatt

Hebb's notion of synaptic modification was not specified completely enough to be simulated or analyzed. Frank Rosenblatt studied a simple neurally-inspired model, called a *perceptron*, for many years, and summarized his work in a 1962 epic, *Principles of Neurodynamics*.

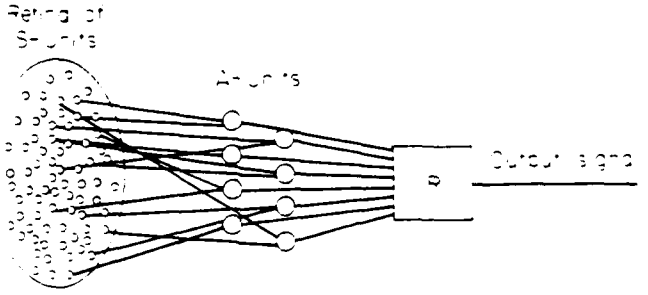

Fig. 3 An elementary perceptron, which consists of a feedforward network from a set ('retina') of input units (S-units) connected with fixed weights to a set of threshold units (A-Units) connected with variable weights to an output unit (R-Units).

Rather than using the fixed weights and thresholds and absolute inhibition of the McCulloch-Pitts neuron, Rosenblatt's units used variable weights with relative inhibition. A perceptron consisted of many such units arranged into a network with some fixed and some variable weights. Fig. 3 shows a typical elementary perceptron. Usually used for pattern-recognition tasks such as object classification, an elementary perceptron consisted of a 'retina' of binary inputs, a set of specific feature detectors, and a response unit. The weights from the input to the middle layer were fixed for an application, and the weights from the detectors to the response unit were iteratively adjusted. The major results of Rosenblatt's work were procedures for adjusting these variable weights on various perceptron implementations, conditions of existence for classification solutions, and proofs that these procedures, under the right conditions, converged in finite time. One statement of the famous 'perceptron convergence theorem' from Rosenblatt is as follows:

> 'Given an elementary α-perceptron, a stimulus world W, and any classification $C(W)$ for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in finite time; then beginning from an arbitrary initial state, an error correction procedure (quantized or non-quantized) will always yield a solution to $C(W)$ in finite time, with all signals to the R-unit having magnitude at least equal to an arbitrary quantity $\delta \geq 0$.'[5]

A world consisted of a set of input patterns to the retina, and a classification was a separation of this world into positive and negative classes. The existence of a guaranteed convergence procedure was very useful; the rub was that the kinds of classifications 'for which a solution exists' were extremely limited. As a footnote to the somewhat incomprehensible proof of this theorem, Rosenblatt attacked a shorter alternative proof by Seymour Papert; an attack, we are sure, he eventually regretted.

2. Symbolic Seventies: paying the price

Many of the early workers of the field were given to extravagant or exuberant claims or overly ambitious goals. McCulloch & Pitts, for example, asserted that 'specification of the net would contribute all that could be achieved in [psychology]'. Ashby clearly overestimated the power of his homeostat, and Rosenblatt stated at a 1958 symposium that:

> [..] it seems clear that the Class C perceptron introduces a new kind of information processing automaton: for the first time we have a machine which is capable of having original ideas'.[6]

He made other dubious claims of power for his perceptron as well, which undoubtedly provoked a backlash. Discussions of this controversy can be found in (Rumelhart & Zipser, 1986) or (Dreyfus & Dreyfus, 1988), and some interesting perspectives on some of the personalities involved can be found in Chapter 4 of McCorduck (1979).

Minsky & Papert

> 'I was trying to concentrate on a certain problem but was getting bored and sleepy. Then I imagined that one of my competitors, Professor Challenger, was about to solve the same problem. An angry wish to frustrate Challenger then kept me working on the problem for a while'.[7]

In 1969, Marvin Minsky and Seymour Papert published *Perceptrons*, a tract which sounded the deathbell for research on perceptrons and other related models. A thoroughgoing mathematical analysis of linear threshold functions showed the limitations of perceptrons both as pattern-recognition machines and as general computational devices.

This book will probably stand permanently as one of the most important works in the field of connectionism, so it is important to understand some of the findings of Minsky & Papert.

First, they defined the *order* of a predicate as the size of the largest conjunction in the minimal sum-of-products logical form for that predicate (or its inverse). Thus, while both conjunction and alternation are predicates of order 1, exclusive-or is a predicate of order 2. The generalization of exclusive-or to more than 2 inputs is parity, which is not of finite order: a sum of products to represent parity of n inputs has at least one term of size n. As a predicate of non-finite order is scaled, then, there is no limit to the necessary fan-in of units, and perceptrons lose their nice aspect of locality.

Using arguments of symmetry, Minsky & Papert then showed, with their *Group Invariance Theorem*, that linear threshold functions which are invariant under a permutation group can be transformed into a function whose coefficients depend only on the group: a major result is that the only linear (i.e. order 1) functions invariant under such transitive groups as scaling, translation and rotation are simple size or area measures. Attempts to use linear functions for, say, optical character recognition under these transitive conditions are thus doomed to failure.

After a cataloguing of the orders of various geometric functions, Minsky & Papert focused on the problems of learning. They showed that as various predicates scale, the sizes of coefficients can grow exponentially, thus leading to systems of impractical memory requirements needing unbounded cycles of a convergence procedure. As Rumelhart & Zipzer pointed out in their review of the perceptron controversy:

> 'The central theme of [*Perceptrons*] is that parallel recognizing elements, such as perceptrons, are beset by the same problems of scale as serial pattern recognizers. Combinatorial explosion catches you sooner or later, although sometimes in different ways in parallel than in serial.'[8]

Minsky & Papert worked on the problem of perceptrons for quite a long time, the result being a boring and sleepy decade for neurally-inspired modeling.

Everybody else

Despite the herbicidal effect of *Perceptrons* on neural network research funding and the flowering of symbolic AI, some research efforts continued to grow during the 70s. Neural network researchers just could not easily publish their work in the AI journals or conferences.

A lot of the work dealt with associative or content addressable memories. Though beyond the scope of this history, significant developments and analyses can be found in the works of Teuvo Kohonen (Kohonen, 1977; Kohonen *et al.*, 1981) and David Willshaw (Willshaw, 1981).

Anderson *et al.* (1977) described experiments with a saturating linear model for pattern association and learning called the 'Brain-State in a Box' or BSB model. Given the current state of the system as a vector, $\vec{X}(t)$ and a matrix of weights, W, the next state of the system can be computed as the inner product between the state and weights, bounded between -1 and 1:

$$\vec{X}(t+1)=\min(1,\max(-1,\vec{X}(t)+W\cdot\vec{X}(t)))$$

Under this system, the state of the system is always within an n-dimensional hypercube (i.e. a 'box') centred around the origin. Anderson was able to apply a type of Hebbian associative learning rule to find weights for this system. BSB models are still being used productively, for example, in the lexical access model of (Kawamoto, 1985).

It is almost impossible to quantify the huge contribution of Stephen Grossberg to neural modeling. The scholarly output of Grossberg and his colleagues at Boston University's Center for Adaptive Systems throughout the seventies is daunting in its mathematical sophistication. Though no excuse, this might account for the allegedly poor scholarship on the part of modern connectionists:

> Rumelhart & Zipser's] discussion does not, however, acknowledge that both the levels and the interactions of a competitive learning model are incompatible with those of an interactive activation model (Grossberg, 1984). The

> authors likewise do not state that the particular competitive learning model which they have primarily analyzed is identical to the model introduced and analysed in Grossberg (1976a, 1976b), nor that this model was consistently embedded into an adaptive resonance model in Grossberg (1976c) and later developed in Grossberg (1978) to articulate the key functional properties [of interactive activation] which McClelland & Rumelhart (1981) describe ...'[9]

It is, of course, possible that the Connectionism of the 80s might in the future be seen as 'Grossberg: Rediscovered'.

3. Exuberant Eighties: research reborn

Interest in connectionist modelling has been on the rise in the 1980s. Perhaps the limits of the symbolic paradigm were beginning to show, perhaps the question of how to program parallel computers became more relevant as their construction became cost-effective, perhaps some agency simply began funding neural models, or perhaps it was simply the ebb and flow of scientific interest. Whatever the reason, the rebirth is now in full swing. This section reviews some of the highlights of recent connectionist history.

Interactive activation

UCSD's Center for Human Information Processing, one of the nation's leading centers for cognitive science, was a staunch supporter of the symbolic paradigm in information-processing psychology. With the publication of *Explorations in Cognition* in 1974, David Rumelhart and Don Norman laid out a research programme strictly in line with the main elements of AI of the time. Propositions, Procedures, Semantic Networks, and Augmented Transition Networks were all used in service of a theory of psychology, and actual computer programs were built which supported the theory.

In 1980, a pair of curious reports were issued from the center 'An interactive activation model of the effects of context in perception, parts 1 and 2' by David Rumelhart and James McClelland (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982). Gone was the link to mainstream AI. Instead, there were 'neuron-like' units, communicating through spreading activation and lateral inhibition. Basically a very small model for explaining many well-known psychological effects of letter recognition in the context of words, their interactive activation model was one of the first high-profile successful applications of modern connectionism.

McClelland & Rumelhart's system, which simulated reactions to visual displays of words and nonwords, dealt only with 4-letter words, and was organized into three distinct levels, **word**, **letter**, and **feature**. The word level contained a group of 1179 units, one for each word, the letter level contained four groups of 26 units each, and the feature level contained four groups of 12 units each for stylized visual features of letters. The system operated by providing input to visual features in all

four letter positions: this activation caused activation in various letter units, which, in turn, caused the activation of possible words. Each group of letter units, and the group of word units, formed what are now called 'winner-take-all' networks, by being fully connected with lateral inhibition links, so that a single unit would tend to dominate all others. Finally, the word units gave positive feedback to their corresponding four letter units.

Clearly in the class of programmed, as opposed to trained, neural network models, Rumelhart & McClelland avoided the morass of individually assigning weights by using uniform weighting schemes for each class of links.

They also provided a justification for the constraints on their model, a justification which neatly sidesteps any claim of neural reality that could open up a philosophical can of worms:

> 'We have adopted the approach of formulating the model in terms which are similar to the way in which such a process might actually be carried out in a neural or neural-like system. We do not mean to imply that the nodes in our system are necessarily related to the behavior of individual neurons. We will, however, argue that we have kept the kinds of processing involved well within the bounds of capability for simple neural circuits.'[10]

The clarion call

In 1982, Jerry Feldman & Dana Ballard published 'Connectionist Models and their Properties', a focusing paper which helped to legitimize connectionism as a methodology for AI and cognitive science. Drawing on both their own work in vision and related neurally-inspired models such as the Rumelhart & McClelland work mentioned above, they sought to unify several strands of research in different fields and define (and name) the bandwagon.[11] Their justifications for abandoning symbolic AI and taking up connectionism were fourfold. First, animal brains are organized differently than computers. Second,

> 'Neurons whose basic computational speed is a few milliseconds must be made to account for complex behaviors which are carried out in a few hundred milliseconds. This means that *entire complex behaviors are carried out in less than a hundred time steps.*'[12]

Third, by studying connectionism we may learn ways of programming the massively parallel machines of the future. And, fourth, many possible mechanisms underlying intelligent behavior cannot be studied within the symbolic programming paradigm.

Feldman & Ballard painted the possibilities of parallelism with broad brushstrokes. Using a framework which included both digital and analog computation, they offered up a large bag of tricks including both primitives for constructing systems (Winner-Take-All Networks, and Conjunctive Connections) and organizing principles to avoid the inevitable combinatorial explosion (Functional Decomposition, Limited Precision Computation, Coding, and Tuning). Although their paper was sprinkled with somewhat fanciful examples, the successful application

of their 'tricks' can be seen in several of the dissertations produced by their students (Cottrell, 1985b; Sabbah, 1982; Shastri, 1985).

Hopfield nets

One of the interesting sociological aspects of the rebirth of connectionism is that valuable contributions are being made from areas other than computer science and psychology. There are several ideas from physics which have entered into the discussion, and perhaps the most notable contributions have come from J.J. Hopfield. He laid out a system for building associative memories based on an analogy to a well-studied physical system, spin glasses (Hopfield, 1982) in which he showed that, by using an asynchronous and stochastic method of updating binary activation values, local minima (as opposed to oscillations) would reliably be found by his method:

> 'Any physical system whose dynamics in phase space is dominated by a substantial number of locally stable states to which it is attracted can therefore be regarded as a general content-addressable memory. The physical system will be a potentially useful memory, if, in addition, any prescribed set of states can readily be made the stable states of the system.'[13]

Hopfield devised a novel way of 'bulk programming' a neural model of associative memory by viewing each memory as a local minimum for a global 'energy' function. A simple computation converted a set of memory vectors into a symmetric weight matrix for his networks.

In a later paper (Hopfield & Tank, 1985) he extended his technique of bulk programming of weights to analog devices and applied it to the solution of optimization problems, such as the NP-complete Travelling Salesman Problem. By designing an energy function whose local minima (or 'attractor states') corresponded to good circuits for a particular configuration of cities, Hopfield's network could rapidly find a reasonably good solution from a random initial state. It should be noted that Hopfield's motivation was **not** to suggest the possibility that $P=NP$ nor to introduce a new approximate algorithm for NP-complete problems, but to demonstrate the usefulness of his neural networks for the kinds of problems which may arise for 'biological computation', and understanding of which may 'lead to solutions for related problems in robotics and data processing using non-biological hardware and software'.[14]

Hopfield has become the symbolic founding father of a very large and broad physics-based study of neural networks as dynamical systems, which is beyond the scope of this survey.

Born-again perceptrons

Of the extension of perceptron learning procedures to more powerful, multilayered systems, Minsky & Papert said:

> 'We consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension is sterile. Perhaps some powerful con-

vergence theorem will be discovered, or some profound reason for the failure to produce an interesting 'learning theorem' for the multilayered machine will be found'.[15]

In the past few years, however, several techniques have appeared which seem to hold the promise for learning in multilevel systems. These are (1) Associative Reward-Penalty, (2) Boltzmann Machine learning, and (3) Back Propagation.

Associative reward-penalty

Working with the goal-seeking units of Klopf (1982), Andrew Barto and colleagues published results in 1982 on one of the first perceptron-like networks to break the linear learning barrier (Barto *et al.*, 1982). Using a two-layered feed-forward network they demonstrated a system which learned to navigate towards either of two locational goals in a small landscape. They showed that in order to have done this successfully, the system had to essentially learn exclusive-or, a nonlinear function.

The task was posed as a control problem for a 'simple organism': At any time t the input to the network was a 7-element vector indirectly indicating location on a two dimensional surface. The output of the network was 4 bits indicating which direction to move (i.e. north, east, south or west). A reinforcement signal, broadcast to all units, based on the before after difference in distance from the goals, was used to correct the weights.

The network had 8 'hidden' units interposed between the 7 input units and 4 output units. One of the factors contributing to the success of their method was that instead of the hidden layer computing binary thresholds, as in an elementary perceptron, it computed positive real numbers, thus allowing gentler gradients for learning.

This early work, on a specific network with a few quirks, was subsequently developed into a more general model of learning, the *Associative Reward-Penalty* or A_{R-P} algorithm. See Barto (1985) for an overview of the work.

Boltzmann Machines

> **Anneal** — To toughen anything, made brittle from the action of fire, by exposure to continuous and slowly diminished heat, or by other equivalent process.
> 'You have been wasted one moment by the vertical rays of the sun and the next annealed hissing hot by the salt sea spray.'[16]

Another notion from physics which has been ported into connectionism is *simulated annealing*. Based on the work of Kirkpatrick *et al.* (1983), Ackley *et al.* (1985) devised an iterative connectionist network which relaxes into a global minimum. As mentioned earlier, Hopfield (1982) constructed a network for associative memory (in which each memory was a local minimum) and showed that an asynchronous update procedure was guaranteed to find local minima. By utilizing a simulated annealing procedure, on the other hand, a Boltzmann machine[17] can find a global minimum.

Given a set of units, s_i, which take on binary values, connected with symmetric weights, w_{ij}, the overall 'energy' of a particular configuration is:

$$E = -\sum_{i<j} w_{ij}s_is_j + \sum_i \theta_i s_i$$

where θ_i are thresholds. A local decision can be made as to whether or not a unit should be on or off to minimize this energy. If a unit is off (O), it contributes nothing to the above equation, but if it is on (1), it contributes:

$$\Delta E_i = \sum_j w_{ij}s_j - \theta_i \ .$$

In order to minimize the overall energy, then, a unit should turn on if its input exceeds its threshold and off otherwise.

But because of the interaction of all the units, a simple deterministic or greedy algorithm will not work. The Boltzmann machine used a stochastic method, where the probability of a unit's next state being on is:

$$\frac{1}{1+e^{-\Delta E_i/T}}$$

where T is a global 'temperature' constant. As this temperature is lowered toward 0, the system state freezes into a particular configuration:

> 'At high temperatures, the network will ignore small energy differences and will rapidly approach equilibrium. In doing so, it will perform a search of the coarse overall structure of the space of global states, and will find a good minimum at that coarse level. As the temperature is lowered, it will begin to respond to smaller energy difference and will find one of the better minima within the coarse-scale minimum it discovered at high temperature.'[18]

To use simulated annealing as an iterative activation function, some units must be 'clamped' to particular states, and a 'schedule' of temperatures and times is used to drive the system to 'equilibrium'. This type of relaxation has been used in two parsing models so far by Selman (1985) and Sampson (1986), and is a computational primitive in the connectionist production system of Touretzky & Hinton (1985).

The real beauty of the Boltzmann machine comes through in its very simple learning rule. Given a desired set of partial states to learn and an initial set of weights, the learning procedure, using only local information, can adjust the weights interactively. By running the annealing procedure several times while clamping over the learning set and several times without any clamping, statistical information about how to change all the weights in the system can be gathered. With slow annealing schedules, their procedure can learn codings for hidden units, thus overcoming some of the limitations of perceptrons.

The down-side of all this is that the learning algorithm is very slow and computationally expensive. Learning a set of weights for a problem may take only hundreds of iterations — but each iteration, in order to collect the statistical information, consists of several trials of simulated annealing, possibly with gentle schedules of thousands of temperatures, for each test case.

Back-propagation

A more robust procedure for learning in multiple levels of perceptron-like units was independently invented and reinvented by several people. In 1981, David Parker apparently disclosed self-organizing logic gates to Stanford University with an eye towards patenting (Parker, 1985); Parker also recently discovered that Paul Werbos developed it in a 1974 mathematics thesis from Harvard University. Yann Le Cun (1985) described a similar procedure in French, and Rumelhart, Hinton & Williams (1986) reported their method, finally, in English.

Perceptrons were linear threshold units in two layers: The first layer detects a set of features, which were hard-coded; the second layer linearly combined these features and could be trained. Convergence procedures for perceptrons would only work on one layer, however, which, among other problems, severly limited their usefulness.

One explanation for why learning could not be extended to more than a single layer of perceptrons is that because of the discontinuous binary threshold, a small change in a weight in one layer could cause a major disturbance for the weights in the next.

By 'relaxing' from a binary to a continuous, analog threshold, then, it is possible to change weights slowly in multiple levels without causing any major disturbances. This is at the basis of the back-propagation technique.

Given a set of inputs, x_i, a set of weights, w_i and a threshold, θ, a threshold logic unit will return 1 if:

$$(\sum_i x_i w_i) - \theta > 0$$

and 0 otherwise. The units used by the back-propagation procedure return:

$$\frac{1}{1 + e^{\theta - \sum_i x_i w_i}}$$

Graphs of these two functions are depicted in Fig. 4. It can be seen that for the analog case, small changes in the input (by changing weights slowly) cause correspondingly small changes in the output.

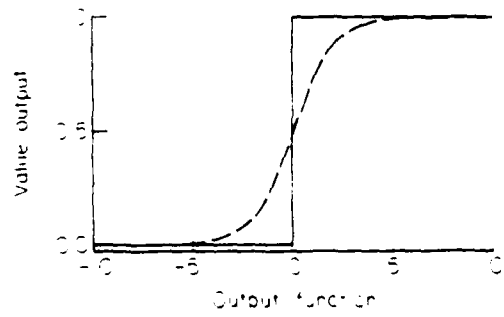

Fig. 4 Graphs of Binary (—) versus Analog (- - -) thresholding functions. The horizontal axis represents the linear combination of inputs and weights minus the threshold, and the vertical axis shows the output function.

Back-propagation has been used quite successfully. Sejnowski & Rosenberg (1986) reported a text-to-speech program which was trained from phonetic data, and Hinton (1986) showed that, under proper constraints, back-propagation can develop semantically interpretable 'hidden' features.

In fact, back-propagation is so widely being used today, that it is threatening to become a subfield of its own. One of the major foci of current connectionism is the application of back-propagation to diverse areas such as sonar (Gorman & Sejnowski, 1988), speech (Elman & Stork, 1987), machine translation (Allen, 1987), and the invention and investigation of numerous tweaks and twiddles to the algorithm (Cater, 1987; Dahl, 1987; Stornetta & Huberman, 1987).

It is for future history to judge whether these new approaches to learning in multiple layers is more than a local maximum in the hill-climbing exercise known as science.

4. Facing the future: problems and prognostications

Because of the problems to be described below, I cannot say with conviction that connectionism will solve major problems for Artificial Intelligence in the near future. I do not believe that the current intense military and industrial interest in neural networks will pay off on a grander scale than did the earlier commercialization of expert systems.

I do believe, however, that connectionism will eventually make a great contribution to AI, given the chance. Its own problems need to be solved first.

Problems for connectionism

Despite the many well-known promises of connectionism, including massively parallel processing, machine learning, and graceful degradation, there are many limitations as well, which derive from naive applications of paradigmatic constraints derived from what is *almost* known about networks of real neurons. Many of these problems only arise when connectionism is applied to higher-level cognitive functions such as natural language processing and reasoning. These problems have been described in various ways, including: recursion, variable-binding, and cross-talk, but they seem to be just variations on older problems, for which entire fields of research have been established.

Generative capacity. Despite the promises of connectionism, the paradigmatic assumptions lead to language processing models which are strictly finite-state. Several parsers have been built which parse context-free grammars of bounded length, i.e. regular grammars. The term 'generative capacity' is due to Chomsky, who used it as a measure of the power (capacity) of particular classes of formal grammars to generate natural language sentences; regular grammars are the weakest in this respect.

For example, as an adjunct to his model for word-sense disambiguation, Cottrell (1985a) proposed a fixed-structure local connectionist model for length-bounded syntactic processing.

In a well-circulated report, Fanty (1985) describes the automatic construction of a connectionist network which parses a context-free grammar. Essentially a time-for-space tradeoff, his system can parse bounded-length sentences, when presented all lexical items at once. The number of units needed for his network to parse sentences of length n rises as $O(n^3)$.

Selman (1985) also reports an automatic construction for networks which can parse bounded-length context-free grammars. His system is stochastic, and based on the Boltzmann machine notions of Ackley *et al.* (1985). Again we have a machine for sentences of bounded length. Another feature of Selman's system is that the connectionist constraint of limited processing cycles is ignored and a parse may take several thousand cycles of annealing.

And even the newer crop of research in this area suffers from the same fixed-width problem (McClelland & Kawamoto, 1986; Allen, 1987; Hanson & Kegl, 1987).

Representational adequacy

Closely related to the problem of generative capacity is the problem of representational adequacy. One must be careful that a model being proposed can actually represent the elements of the domain being modeled. One of the major attacks on connectionism has been on the inadequacy of its representations, especially on their lack of compositionality (Fodor & Pylyshyn, 1988). In feature-based distributed representations, such as the one used by Kawamoto (1985), if the entire feature system is needed to represent a single element, then attempting to represent a structure involving those elements cannot be managed in the same system. For example, if all the features are needed to represent a *nurse*, and all the features are needed to represent an *elephant*, then the attempt to represent a *nurse riding an elephant* will come out either as a *white elephant* or a rather *large nurse with four legs*.

One obvious solution to this problem of superimposition versus concatenation involves using separate 'pools' of units to represent elements of propositional triples, such as Agent, Action, and Object. In each pool would reside a distributed representation filling these roles such as 'Nurse', 'Riding', and 'Elephant'. Because of the dichotomy between the representation of a structure (by concatenation) and the representation of an element of the structure (by features), this type of system cannot represent recursive propositions such as 'John saw the nurse riding an elephant'.

Finally, parallel representations of sequences which use implicit sequential coding (such as Rumelhart & McClelland (1986) used in their perceptron-like model for learning the past tenses of verb) have limits representing repetitive constituents. So a system, for example, which represented words as collections of letter-triples, would not be able to represent words with duplicate triples such as *banana*.

Task control

A final problem is that many neural models use every 'allowable' device they have to do a single task, this leaves no facility for changing tasks, or even changing the

size of tasks, except massive duplication and modification of resources. For example, in the past-tense model (Rumelhart & McClelland, 1986), there is no obvious means to conjugate from, say, past to present tense, without another 200,000 weights. In the Travelling Salesman network (Hopfield & Tank, 1985), there is no way to add a city to the problem without configuring an entire new network.

Predicting the future

The existence and recognition of these problems is slowly causing a change in the direction of near-term connectionist research. There are many ongoing efforts now on more serial approaches to recognition and generation problems (Elman, 1988; Gasser & Dyer, 1988; Jordan, 1986; Pollack, 1987), which may help overcome the problem of massive duplication in dealing with time. There is also research in progress along the lines of Hinton's (1988) proposal for reduced descriptions as a way out of the superposition concatenation difficulty for distributed representations. For example, Pollack (1988) demonstrates a reconstructive distributed memory for variable sized trees, and Dyer *et al.* (1988) show a network construction for representing simple semantic networks as labelled directed graphs.

As problems in capacity, representation, and control are solved, we may expect a new blooming of connectionist applications in areas currently dominated by traditional symbolic processing.

I believe that connectionism may lead to an implementational redefinition of the notion of 'symbol'. In AI, symbols have no internal structure and thus mean very little: they are just used as names for, or pointers to, larger structures of symbols, which are reasoned with (slowly). The essential difference between the early neural network research and modern connectionism is that AI has happened in-between them. Because modern connectionism really does focus on representations, there is a possibility that a new kind of symbol might emerge from connectionism. For example, a reduced representation of some structure into a distributed pattern could be considered such a symbol, given that it can 'point' to a larger structure through a reconstruction algorithm. Such 'supersymbols' as opposed to subsymbols (Smolensky, 1988) may have an advantage over AI style token-symbols, in that they possess internal structure which can be reasoned about.

Finally, I wish to make quite a far-fetched prediction, which is that Connectionism will sweep AI into the current revolution of thought in the physical and biological sciences (Crutchfield *et al.*, 1986; Gleick, 1987; Grebogi *et al.*, 1987). Fig. 5 shows a set of disciplines which are almost communicating today, and implies that the shortest path between AI and chaos is quite long.

There has already been some intrusion of interest in chaos in the physics-based study of neural networks as dynamical systems. For example, both Huberman & Hogg (1987) and Kurten (1987) show how phase-transitions occur in particular neural-like systems, and Lapedes & Farber, (1988) demonstrates how a network trained to predict a simple iterated function would follow that function's bifurcations into chaos. However, these efforts are strictly bottom-up and it is still difficult to see how chaos has anything to do with connectionism, let alone AI.

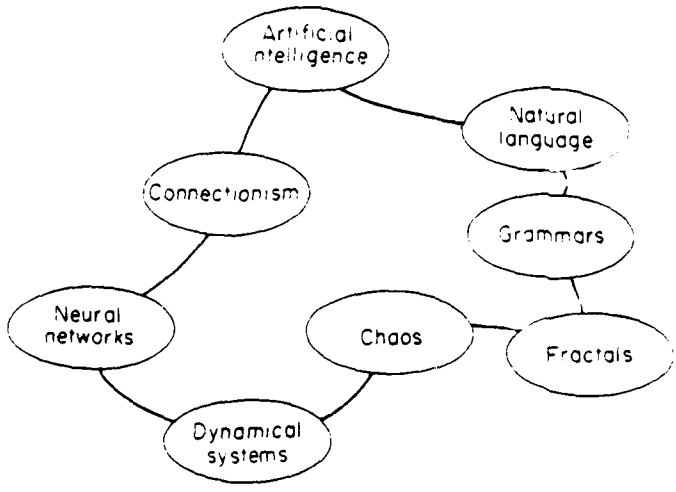

Fig. 5 Research areas which almost communicate.

Taking a more top-down approach, consider several problems which have been frustrating for some time. One problem is how to get infinite generative capacity into a system with finite resources (i.e. the competence performance distinction). Another is the question of reconstructive memory, which has only been crudely approximated by AI systems (Dyer, 1983). Yet another is the symbol-grounding problem, which is how to get a symbolic system to touch ground in real-world perception and action, when all systems seem to bottom out at an a priori set of semantic primitives.

My suspicion is that many of these problems stem from a tacit acceptance, by both AI researchers and connectionists, of 'Aristotelian' notions of knowledge representation, which stop at terms, features, or relations. Just as Mandelbrot claims to have replaced the ideal integer-dimensional Euclidean geometry with a more natural fractional dimensional (fractal) geometry (Mandelbrot, 1982) so we may ultimately have to create a non-Aristotelian representational base.

I have no concrete idea on what such a substrate would look like, but consider something like the Mandelbrot set as the basis for a reconstructive memory. Nearly everyone has seen glossy pictures of the colourful shapes that recurrently appear as the location and scale are changed. Imagine an 'inverse' function, which, given an underspecified picture, quickly returns a 'pointer' to a location and scale in the set. Reconstructing an image from the pointer fills in the details of the picture in a way consistent with the underlying self-similarity inherent in the memory. Given that all the representations to be 'stored' are very similar to what appears in the set, the ultimate effect is to have a look-up table for an infinite set of similar representations which incurs no memory cost for its contents. Only the pointers and the reconstruction function need to be stored.

While it is not currently feasible, I think that approaches like this to reconstructive memory may also engender systematic solutions to the other problems, of finitely regressive representations which bottom out at perception rather than at primitives, and which give the appearance of infinite generative capacity.

5. Conclusion

Like many systems considered historically, connectionism seems to have a cyclical nature. It may well be that the current interest dies quite suddenly due to the appearance of another critical tour-de-force such as *Perceptrons*, or a major accident, say, in a nuclear power plant controlled by neural networks. On the other hand, some feel that AI is entering a retrenchment phase, after the business losses recently suffered by its high-profile corporate entities and the changing of the guard at DARPA. Given that it doesn't all go bust, I predict that the current limitations of connectionism will be understood and or overcome shortly and that, within 10 years, 'connectionist fractal semantics' will be a booming field.

Notes

McCulloch & Pitts, 1943, p. 124.

Hebb, 1949, p. xiv, emphasis mine.

Ibid., p. 62.

Ashby, 1960, p. 10).

Rosenblatt, 1962, p. 111).

Rosenblatt, 1959, p. 449). This type of claim has not been repeated in AI history until the advent of 'Discovery systems' (Lenat, 1977).

Minsky, 1986, p. 42).

Rumelhart & Zipser, 1986, p. 158).

Grossberg, 1987, pp. 27–28).

McClelland & Rumelhart, 1981, p. 387).

'Connectionism' was the name of some very antiquated psychological theory that even Hebb alluded to. And though the Rumelhart School has tried to rename their work as 'Parallel Distributed Processing' or 'PDP' models, it seems that 'Connectionism' has stuck. But new names keep appearing, such as Neurocomputing, Neuroengineering, and Artificial Neural Systems.

Feldman & Ballard, 1982, p. 206).

Hopfield, 1982, p. 2554).

Hopfield & Tank, 1985, p. 142).

Minsky & Papert, 1969, p. 232).

Definition from the compact edition of the Oxford English Dictionary, their citation from Michael Scott, *Cringle's Log*, 1859.

No hardware actually exists. The name is an honour, like Von Neumann machine.

Ackley, *et al*., 1985, p. 152).

References

Ackley, D. H., Hinton, G. E. & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann Machines. *Cognitive Science*, **9**, 147–169.

Allen, R. (1987). Several studies on natural language and back propagation. In *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*. San Diego, Vol. II, 335–342.

Anderson, J. A., Silverstein, J. W., Ritz, S. A. & Jones, R. S. (1977). Distinctive Features, Categorical Perception, and Probability Learning: Some Applications of a Neural Model. *Psychological Review*, **84**, 413–451.

Ashby, W.R. (1960) *Design for a Brain: The origin of adaptive behaviour.* (2nd edn.) John Wiley & Sons, New York.

Barto, A.G., Anderson, C.W. & Sutton, R.S. (1982) Synthesis of Nonlinear Control Surfaces by a layered Associative Search Network. *Biological Cybernetics*, **43**, 175–185.

Barto, A.G. (1985) Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, **4**, 229–256.

Cater, J.P. (1987). Successfully using peak learning rates of 10 (and greater) in back-propagation networks with the heuristic learning algorithm. In *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*. San Diego, II-645-652.

Cottrell, G.W. (1985a) Connectionist Parsing. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*. Irvine, CA, USA.

Cottrell, G.W. (1985b) *A Connectionist Approach to Word-Sense Disambiguation*. TR154 Computer Science Department, University of Rochester, Rochester, USA.

Crutchfield, J.P., Farmer, J.D., Packard, N.H. & Shaw, R.S. (1986) Chaos. *Scientific American*, **255**, 46–57.

Cun, Y. Le (1985) A Learning Scheme for Asymmetric Threshold Networks. In *Proceedings of Cognitiva 85*, Paris, 599–604.

Dahl, E.D. (1987) Accelerated learning using the generalized delta rule. In *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*, San Diego, II-523-530.

Dreyfus, H.L. & Dreyfus, S.E. (1988) Making a Mind versus Modeling the Brain: Artificial Intelligence Again at the Crossroads. *Daedalus*, **117**, 15–43.

Dyer, M.G. (1983). *In Depth Understanding*. MIT Press, Cambridge, USA.

Dyer, M.G., Flowers, M. & Wang, Y.A. (1988) Weight Matrix = Pattern of Activation: Encoding Semantic Networks as Distributed Representations in DUAL, a PDP architecture. *UCLA-AI-88-5*. Los Angeles: Artificial Intelligence Laboratory, UCLA.

Elman, J. & Stork, D. (1987) Session on Speech Recognition and Synthesis. In *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*, San Diego, IV-381-504.

Elman, J.L. (1988) *Finding Structure in Time*. Report 8801, San Diego: Center for Research in Language, UCSD.

Fanty, M. (1985) *Context-free parsing in Connectionist Networks*. TR174, Rochester, N.Y., Computer Science Department, University of Rochester, USA.

Feldman, J.A. & Ballard, D.H. (1982) Connectionist models and their properties. *Cognitive Science*, **6**, 205–254.

Fodor, J. & Pylyshyn, A. (1988) Connectionism and Cognitive Architecture: A Critical Analysis. *Cognition*, **28**, 3–71.

Gasser, M. & Dyer, M.G. (1988) Sequencing in a Connectionist Model of Language Processing. In *Proceedings of the 12th International Conference on Computational Linguistics*. Budapest.

Gleick, J. (1987) *Chaos: Making a new science*. Viking, New York.

Gorman, R.P. & Sejnowski, T.J. (1988) Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, **1**, 75–90.

Grebogi, C., Ott, E. & Yorke, J.A. (1987) Chaos, Strange Attractors, and Fractal Basin Boundaries in Nonlinear Dynamics. *Science*, **238**, 632–638.

Grossberg, S. (1987) Competitive Learning: From Interactive Activation to Adaptive Resonance. *Cognitive Science*, **11**, 23–63.

Hanson, S.J. & Kegl, J. (1987) PARSNIP: A connectionist network that learns natural language grammar from exposure to natural language sentences. In *Proceedings of the Ninth Conference of the Cognitive Science Society*. Seattle, 106–119.

Hebb, D.O. (1949) *The Organization of Behavior: A Neuropsychological Theory*. John Wiley & Sons, New York.

Hinton, G.E. (1986) Learning Distributed Representations of Concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*. Amherst, MA, 1–12.

Hinton, G.E. (1988) Representing part-whole hierarchies in connectionist networks. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*. Montreal, p. 48–54.

Hopfield, J.J. (1982) Neural Networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, **79**, 2554–2558.

Hopfield, J.J. & Tank, D.W. (1985) 'Neural' computation of decisions in optimization problems. *Biological Cybernetics*, **52**, 141–152.

Huberman, B.A. & Hogg, T. (1987) Phase Transitions in Artificial Intelligence Systems. *Artificial Intelligence*, **33**, 155–172.

Jordan, M.I. (1986) *Serial Order: A Parallel Distributed Processing Appraoch*. ICS report 8608. La Jolla, Institute for Cognitive Science, UCSD.

Kawamoto, A.H. (1985) *Dynamic Processes in the (Re)Solution of Lexical Ambiguity*. Doctoral Dissertation, Providence: Brown University, Department of Psychology, USA.

Kirkpatrick, S., Gelatt, C.D. & Vecchi, M.P. (1983) Optimization by simulated annealing. *Science*, **220**, 671–680.

Klopf, A.H. (1982) *The Hedonistic Neuron*. Hemisphere Publishing Corporation, Washington, DC.

Kohonen, T. (1977) *Associative Memory: A Systems-Theoretical Approach*. Springer-Verlag, Berlin.

Kohonen, T., Oja, E. & Lehtio, P. (1981) Storage and Processing of Information in Distributed Associative Memory Systems. In G.E. Hinton & J.A. Anderson, (Eds.), *Parallel models of associative memory*. Lawrence Erlbaum Associates, Hillsdale, USA.

Kurten, K.E. (1987) Phase transitions in quasirandom neural networks. In *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*. San Diego, II-197-204.

Lapedes, A.S. & Farber, R.M. Nonlinear signal processing using several networks: prediction and system modeling. *Biological Cybernetics* (in press).

Lenat, D.B. (1977) The Ubiquity of Discovery. *In Proceedings of the Fifth International Joint Conference on Artifical Intelligence*. Cambridge, MA, 1093–1105.

Mandelbrot, B. (1982) *The Fractal Geometry of Nature*. Freeman, San Francisco.

McClelland, J.L. & Rumelhart, D.E. (1981) An interactive activation model of the effect of context in perception: Part 1. An account of basic findings. *Psychology Review*, **88**, 375–407.

McClelland, J. & Kawamoto, A. (1986) Mechanisms of Sentence Processing: Assigning Roles to Constituents. In J.L. McClelland, D.E. Rumelhart & the PDP research Group, (Eds.), *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*. Vol. 2. MIT Press, Cambridge, USA.

McCorduck, P. (1979) *Machines Who Think*. Freeman, San Francisco, CA, USA.

McCulloch, W.S. & Pitts, W. (1943) A logical calculus of the ideas immanent in nervious activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133.

Minsky, M. & Papert, S. (1969) *Perceptrons*. MIT Press, Cambridge, MA., USA.

Minsky, M. (1986) *The Society of Mind*. Simon & Schuster, New York.

Norman, D.A. & Rumelhart, D.E. (1975) *Explorations in Cognition*. W.H. Freeman & Co., San Francisco.

Parker, D.B. (1985) *Learning Logic*. Technical Report-47, MIT Center for Computational Research in Economics and Management Science, Cambridge, MA, USA.

Pollack, J.B. (1987) Cascaded Back Propagation on Dynamic Connectionist Networks. In *Proceedings of the Ninth Conference of the Cognitive Science Society*. Seattle, 391–404.

Pollack, J.B. (1988) Recursive Auto-Associative Memory: Devising Compositional Distributed Representations. *Proceedings of the Tenth Conference of The Cognitive Science Society*. Montreal, pp. 33–39.

Rosenblatt, F. (1959) Two theorems of statistical separability in the perceptron. In *Mechanization of Thought Processes*. Vol. 1. H.M.S.O., London.

Rosenblatt, F. (1962) *Principles of Neurodynamics*. Spartan, New York.

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986) Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cog-*

nition. Vol. 1 (eds J.L. McClelland, D.E. Rumelhart & the PDP research Group). MIT Press, Cambridge.

Rumelhart, D.E. & McClelland, J.L. (1982) An interactive activation model of the effect of context in perception: Part 2 The contextual enhancement effect and some tests and extensions of the model. *Psychology Review*, **89**, 60–94.

Rumelhart, D.E. & Zipser, D. (1986) Feature Discovery by Competitive Learning. In D.E. Rumelhart, J.L. McClelland & the PDP research Group, (eds.), *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*. Vol. 1. MIT Press, Cambridge, USA.

Rumelhart, D.E. & McClelland, J.L. (1986) On Learning the Past Tenses of English Verbs. In *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*. Vol. 2. J.L. McClelland, D.E. Rumelhart & the PDP research Group, (eds.). MIT Press, Cambridge, USA.

Sabbah, D. (1982) *A Connectionist Approach to Visual Recognition*. TR107 Computer Science Department, University of Rochester, USA.

Sampson, G. (1986) A stochastic approach to parsing. In *Proceedings of the 11th International Conference on Computational Linguistics*, pp. 151–155. Bonn.

Sejnowski, T.J. & Rosenberg, C.R. (1986) *NETtalk: A parallel network that learns to read aloud*. JHU EECS-86 01: The Johns Hopkins University, Electrical Engineering and Computer Science Department.

Selman, B. (1985) *Rule-Based Processing in a Connectionist System for Natural Language Understanding*. CSRI-168, Toronto, Computer Systems Research Institute, University of Toronto, Canada.

Shastri, L. (1985) *Evidential reasoning in semantic networks: A formal theory and its parallel implementation*. TR166, Rochester: Computer Science Department, University of Rochester.

Smolensky, P. (1986) Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Parallel Distributed Processing: Experiments in the Microstructure of Cognition*. Vol. 1. D.E. Rumelhart, J.L. McClelland & the PDP research Group, (eds.). MIT Press, Cambridge, USA.

Smolensky, P. (1988) On the proper treatment of Connectionism. *Behavioural and Brain Sciences* **II**, no. 1, 1–23.

Stornetta, W.S. & Huberman, B.A. (1987) An Improved three-layer back propagation algorithm. In *Institute of Electrical and Electronics Engineers First International Conference on Neural Networks*. San Diego, II-637-644.

Touretzky, D.S. & Hinton, G.E. (1985) Symbols among the neurons: details of a connectionist inference architecture. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*. Los Angeles, CA, USA.

Willshaw, D.J. (1981) Holography, Associative Memory, and Inductive Generalization. In G.E. Hinton & J.A. Anderson, (eds.), *Parallel models of associative memory*. Lawrence Erlbaum Associates, Hillsdale, USA.

Learning in Parallel Distributed Processing Networks: Computational Complexity and Information Content

JOHN F. KOLEN and ASHOK GOEL

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210, USA

(614)-292-7402

kolen-j@cis.ohio-state.edu

May 31, 1989

Learning in Parallel Distributed Processing Networks: Computational Complexity and Information Content

JOHN F. KOLEN and ASHOK GOEL

Laboratory for Artificial Intelligence Research
Department of Computer and Information Science
The Ohio State University

Abstract

Connectionist networks offer an intriguing set of techniques for learning based on the adjustment of weights of connections between processing units. To more precisely identify the power and limitations of connectionist learning, we conducted a set of experiments on learning using the mechanism of *back propagation* of corrective feedback. The experiment on learning to compute the exclusive-OR function explores the computational efficiency of connectionist learning and suggests that the efficiency is a function of the *initial conditions*. The experiment on learning to play *Tic-Tac-Toe* investigates the information content of what is learned and indicates that the ability to generalize is dependent on the *environmental conditions*. We also provide a formal proof for the computational intractability of learning in connectionist networks. These results strongly suggest that what is needed is a decomposition of the learning space so that network can navigate simpler, smaller spaces more efficiently. The need for decomposing the learning space raises the issue of how to search for the right kind of structure. We propose that one possibility lies in the direction of developing *task-specific* architectures.

Keywords: Back propagation, computational complexity, generalization, information content, neural networksm, parallel distributed processing.

Introduction

Machine learning has long been an important issue in Artificial Intelligence research. As early as the mid-sixties, Samuel (Samuel, 1967) identified three central issues concerning the development of information processing systems capable of learning. First, the system must be able to internally represent what it knows and what is to be learned: *the representation problem*. Second, once incorrect performance has been detected, the system must be able to identify which part of the system is responsible for the incorrect performance (and make modifications to achieve correct performance): *the credit assignment problem*. Third, the system must be capable of generalizing over existing abstractions in its memory: *the generalization problem*. In the late sixties, Minsky and Papert (Minsky and Papert, 1988) emphasized the importance of a fourth constraint, namely, the system must be capable of learning within a reasonable amount of time: *the computational complexity problem*.

Since then, several attempts have been made at building systems that learn. Until recently, these efforts were based largely on algorithmic processes operating on discrete symbolic representations (Carbonell *et al.*, 1983). This research has led to the development of several techniques for learning such as learning from examples, from analogies, from explanations. While this line of research has led to the building of small-scale systems that learn more or less well in relatively narrow domains, computationally feasible solutions to the general problem of learning are yet to be discovered.

Recently, research on neural networks has led to the development of a different set of techniques for learning based on the adjustment of weights of connections between processing units in a "continuous" space. Again, these techniques have been used with some success at building small-scale systems capable of learning relatively simple tasks in narrow domains. However, the limitations of these techniques are not yet entirely clear. If, for instance, a network merely memorizes the correct solutions to the specific exemplars on which it is trained, then there would be little *a priori* reason to believe that it will behave appropriately when presented with a novel situation. An adequate general solution to the learning problem must enable the system to generalize.

To more precisely identify the power and limitations of connectionist learning, we conducted a set of experiments on learning using the mechanism of *back propagation* of corrective feedback.

Elsewhere (Goel *et al*, 1988) we provided a preliminary account of the experimental results concerning the credit assignment part of the learning problem. In this paper, we describe some of the experiments in detail, and discuss the results in relation to the general problem of learning. First, we present a brief review of learning in connectionist networks. Next, we report on a set of experiments on learning to compute the exclusive-OR function. The goal of this experiment is to determine the *initial conditions* under which the back propagation method can learn computationally efficiently. We then report on an experiment on learning to play the game of *Tic-tac-toe*. The aim of this experiment is to identify the information content of what connectionist architectures typically learn, and the effect of *environmental conditions* on this learning. We also discuss the issue of computational complexity of learning, we provide a formal proof for the intractability of learning in connectionist networks. Finally, we discuss potential solutions to some of the problems of learning in connectionist networks..

Learning in Parallel Distributed Processing Networks

Neural networks come in a number of different varieties, e.g. *parallel distributed processing* (PDP) (Rumelhart and McClelland, 1986a), *Hopfield networks* (Hopfield, 1982), and *adaptive resonance theory* (Carpenter and Grossberg 87). Within the PDP framework, a number of learning techniques have been developed, e. g., *back propagation* (Rumelhart *et al.*, 1986a), *harmony theory* (Smolensky, 1986) , and *Boltzman machines* (Hinton and Sejnowski, 1986). In this paper, we confine our attention to the technique of back propagation; however, much of our analysis is relevant to other connectionist learning schemes as well.

The processing units in PDP networks are organized in input, output, and one or more hidden layers. The units in the various layers are connected to each other; Figure 1 shows the connection between units i and j has a weight of w_{ij}. The output of a unit is computed by means of a semi-linear threshold function. A fairly common function of this class is the classic logistic function (Rumelhart *et al.*, 1986a):

$$o_j = \frac{1}{1 + e^{-(\sum_i w_{ij} o_i - \theta)/T}} \qquad (1)$$

where θ is the threshold of the unit, and temperature, T, regulates the slope of the function within the critical region.

Insert Figure 1 about here.

It has been demonstrated that, at least in principle, it is possible to design connectionist networks that are Turing-Universal (McCulloch and Pitts, 1943; Pollack, 1987; Franklin and Garzon 1988). However, in practice, designing a connectionist network to compute a given function can be rather difficult. The critical choices are selecting the right number of layers, the right number of units for each layer, and determining how they should be connected. To see how hard designing connectionist network can be, let us consider the design of a n input, 1 output network. Assuming binary inputs and outputs, the network can compute up to 2^{2^n} functions, each requiring a different weight configuration. If m is the number of connections in the network, and W is the set of possible weights for each connection in the network, then there are $|W|^m$ weight configurations, where $|W|$ is the size of W. To cover the entire function space, W must contain at least $2^{2^n/m}$ distinguishable weights.[1]

Learning in connectionist networks takes the form of adjusting the weights of connections between the processing units in the network. The network is subjected to training during which

- a specific exemplar of the given task is presented as input to the network and the incorrect output of the network is detected,

- a corrective feedback is supplied by the trainer and back propagated to the individual con-

[1] For simplicity we assume that each weight configuration specifies a unique function, however, some functions may be specified by more than one weight configurations.

nections in the network, and

- the connection weights are changed in the direction of steepest gradient descent in the error space.

This procedure, called back-propagation, is often expressed as the *generalized delta rule* (Rumelhart *et al.*, 1986a), which calculates error signals at each unit of the form

$$\Delta w_{ij} = \eta \delta_j o_i$$

with

$$\delta_j = (t_j - o_j)$$

for the output units, and

$$\delta_j = \sum_k \delta_k w_{kj}$$

for the hidden units, where Δw_{ij} is the change in the weight w_{ij} of the connection from unit i to unit j, o_i is the output of the unit i, t_j is the correct output supplied by the trainer, o_j is the output of the network, and η is a constant of proportionality called the learning rate. This procedure is executed repeatedly until some performance criterion is met (e.g., the weights of the connections in the network stabilize), and the network converges to the correct solution for specific examples of the task. In this manner the network is trained to correctly perform the task on a set of specific cases selected by the trainer.

Learning to Compute XOR

The first experiment on learning in PDP networks we conducted was learning to compute the Boolean function of two-input exclusive-OR (XOR). The goal of this experiment is to identify the conditions under which back-propagation of corrective feedback using the generalized delta rule leads to computationally efficient learning. The XOR problem is a simple version of the more general parity problem (Minsky and Papert, 1988) with input patterns of size two. Rumelhart, Hinton and Williams (Rumelhart *et al.*, 1986a) have earlier reported on a PDP network that learns to

compute the XOR function. Their network consists of two units for the two inputs, an output unit for the one output, and one hidden unit. The network learned to correctly compute XOR in a few hundred (558) sweeps through the four stimulus patterns. This implies that starting from an initially random set of the connection weights, it took a few thousand ($558 \times 4 = 2232$) training sessions to adjust the connection weights before the network learned to compute XOR correctly.

As Minsky and Papert (Minsky and Papert, 1988) have pointed out, it is rather difficult to analyze the meaning of these results for the general parity problem without knowing how well the learning scheme performs on input patterns of size greater than two. In fact, it is difficult to evaluate the computational efficiency of the learning scheme even for input patterns of size two since Rumelhart *et al* do not provide much of the needed information. While the learning rule is reported to be the generalized delta rule with a learning rate of $\eta = 0.5$, the range in which the initial weights of the connections can vary is not specified. A specification of the range of weights is important since it defines the size of the parameter space which has to be navigated. Further, no mention is made of whether the biases of the processing units in the network are fixed or can vary in some range. In short, while one may conclude from this experiment that the network learned to compute the XOR function correctly, it is not possible to draw any definitive conclusions about the computational efficiency of learning.

In order to determine the conditions under which this network learns to compute XOR efficiently, we repeated the experiment of Rumelhart *et al*. In our simulation, we used the same network architecture (a schematic of the network is shown in Figure 2). The learning rule (the generalized delta rule) and the learning rate ($\eta = 0.5$) were also kept the same. The range of the initial weights and the biases were 0.0 to 5.0. The simulation was performed a hundred times, the results of which are shown in Figure 3. The x coordinate identifies the number of sessions that were required to train the network, while the y coordinate identifies the number of simulations (out of 100) for which a given number of training sessions were required. The simulations that took more than two hundred thousand training sessions have been collapsed giving the spike at two hundred thousand sessions. In each of the 100 simulations, the network started with a randomly generated initial set of connection weights, and eventually learned the set of weights for correctly computing XOR. We note

that in a few of the simulations we were able to reproduce the results reported by Rumelhart *et al.* However, for a vast majority of simulations it took the network substantially more training sessions before it learned to compute XOR correctly. In fact, for nearly half the simulations this took more than two hundred thousand training sessions, two orders of magnitude higher than those reported by Rumelhart *et al.*

Insert Figure 2 about here.

Insert Figure 3 about here.

The important issue is the differences between conditions under which the network learned to correctly compute XOR in only a few thousand training sessions, and those under which it took a few hundred thousand sessions. Since the training procedure and learning rate are the same, it appears that the difference lies in the initial (random) selection of connection weights. That is the efficiency of the learning scheme depends not so much on the use of the generalized delta rule, as much as on the initial choice of weights.

It could be argued that the above experiment violates the "standard" procedure of starting the initial weights very "close" to zero ($|w| < 0.3$ typically) so that the slope of the output function

is large enough. In anticipation of this objection, we repeated the above experiment, this time varying several parameters in the simulation: initial weight range, learning rate, η, and momentum rate, α. We were interested in the number of initial weight states which did not converge to the target function within 50000 epochs.[2] During the simulation, network outputs less than or equal to 0.49 where considered 0, while outputs greater than or equal to 0.51 were considered 1. Figure 4 illustrates the results of this experiment. In this graph, the x axis represents the range of initial weights which were randomly generated between 0.0 and x, and the y axis represents the percentage of initial weight configurations which did not converge in 50000 epochs. Two conclusions can be drawn from these results. First, learning convergence is more likely if the initial weights are very small. Second, the graph also shows that varying the learning rate (L) and momentum rate (M) had little effect on the percentage of non-convergent initial weights. Similar results were obtained when the absolute value of the initial weights were bounded by a parameter, as shown in Figure 5. In sum, the computational efficiency of learning depends on initial conditions of the network.

Insert Figure 4 about here.

Insert Figure 5 about here.

[2] An epoch is the presentation of the entire exemplar set. In this case, an epoch consisted of four exemplars.

Learning Tic-Tac-Toe

The second problem that we have investigated is learning to play *Tic-tac-toe*, the 3×3 board game in which two players take turns placing distinguishable marks on the board. The goal of this experiment is to identify the content of what connectionist architectures actually learn. In *Tic-tac-toe*, the player who first succeeds in capturing an entire row, column or diagonal wins the game. Against an experienced opponent, typically the best a player can do is to achieve a stalemate. Rumelhart, Smolensky, McClelland and Hinton (Rumelhart *et al.*, 1986b) have earlier reported on a connectionist network that learns to play *Tic-tac-toe*. Their network contains 67 units, including 9 input units for the network's current board position, 9 input units for the opponent's current position, and 9 output units for the next move by the network. There are also 40 hidden units, organized in 8 groups corresponding to the 8 ways in which an entire row, column or diagonal can be captured. Each such group contains 5 units corresponding to the five abstractions of *an empty line*, *friendly singlet*, *friendly doublet*, *opponent singlet*, and *opponent doublet*.

Once again, it is rather difficult to analyze the implications of this experiment for learning in connectionist networks since it is not clear what has the network really learned? What enables the network to select the correct move for a given board position are the information processing abstractions of *a row*, *opponent doublet*, *etc*. These abstractions serve to reduce the already relatively small learning space even further, and guide the network in navigating the reduced learning space. In fact, it is these information processing abstractions that form a theory of how to play the game of *Tic-tac-toe*. However, the network does not learn these abstractions. Instead, the system designers "programmed" these abstractions into the network in "compiled" form (Chandrasekaran *et al*., 1988).

The task of learning to play *Tic-tac-toe* would be much harder if the network lacked the above information processing abstractions to begin with. In fact, the real test of learning would be whether the network can learn these abstractions. Thus, we designed a connectionist network that learns to play *Tic-tac-toe* without providing it with any of these abstractions. The network, shown in Figure 6, contains 9 input units for the network's current board position, 9 input units for the opponents

current position, and 9 output units for the next move, just as in the network of Rumelhart *et al.*[3] The network also contains hidden units, the number of which is kept as a parameter of the network.

Insert Figure 6 about here.

In order to reduce the time required for a training session, we used a variation on the generalized delta rule. For a fixed precision, the continuous and monotonic activation function degenerates to the linear threshold function as temperature T approaches 0, as can be seen from Equation 1 (see section 2). Since the activation function is now a step function, this implies that the derivative of the function would be 0. However, the derivative only adjusts the magnitude of the change in the connection weights, not its direction. Thus, if the steepest gradient descent method is used for calculating the change in connection weights but drop the derivative term from the calculations, then the weights would be changed in the right direction but the actual distance traversed in the error space would typically be somewhat larger. This tends to reduce the time for a training session and also makes the back propagation less sensitive to local perturbations in the error space.

We provided a symbolic algorithm as the opponent in to the connectionist network. The symbolic algorithm used the information processing abstractions that were denied to the connectionist network. The algorithm used a simple heuristic: for any given board position, it selected the move that would maximize the number of rows, columns, and diagonals of which it had sole possession, while minimizing the number of rows, columns, and diagonals possessed by its opponent (the network). Thus, given the same board configuration the symbolic algorithm would always generate the same move. Moreover, because of the nature of the heuristic, there existed a sequence of

[3]The network actually contained an extra input unit for "symmetry breaking"(Goel *et al.*, 1988a), the need for which arises when the board is empty and the network has to make the first move.

moves which could beat the algorithm. This made the symbolic algorithm a strong, yet imperfect, opponent for the network.

In the training procedure we adopted, the network was first made to play the symbolic opponent, and then learn from the moves made by the winning side. Initially, the network played the game by evaluating the current board position and generating a list of the moves it would like to make. This list includes both legal and illegal moves, where an illegal move involves moving into an already occupied square. Illegal moves were then stripped from the list, and the actual move was selected (at random) from the remaining moves.[4] During actual play, a record was kept of the moves throughout the game. At the end of the game, the network was trained to make the same responses to situations encountered by the victorious opponent.[5] This was done by using a variation of the generalized delta rule as described above. Thus, the network started out as a random move generator, and then learned from its opponent each time it (the network) lost a game.

In this manner, the network played the symbolic algorithm, and learned from it, for 10 matches, each match consisting of a 1000 games. The results are shown in Table 1.[6] Each row corresponds to a given match and contains the number of games (out of 1000) that the symbolic algorithm won, the number that the network won, and that ended in a stalemate. (Matches were sorted by the neural networks performance.) We found that in each of the 10 matches the network learned to draw its opponent in about 100 games. We also discovered that although it was possible to beat the imperfect symbolic algorithm, in fact, the network won exactly one game out of 10,000 games.

Insert Table 1 about here.

[4] If the network suggested no moves or only illegal moves, then program randomly selected a legal move.

[5] The network was trained to be the best of the two opponents.

[6] We have not provided here any details about the parameters of the network such as the learning rates, the ranges of the connection weights and the unit biases, since the goal of this experiment was not to measure the computational efficiency of learning, but to investigate the content of what is learned.

To win against the symbolic algorithm consistently would have required that the network learn information processing abstractions such as *an empty line*, *an opponent doublet*, *etc.*, which were "precompiled" into the network of Rumelhart *et al.* However, we could detect no such abstractions despite a careful examination of the activation levels of the hidden units in the network. It appears that the network learned to *mimic* its opponent but could not generalize to the abstractions needed to win against the opponent. The ability to generalize to abstractions is influenced by the *environmental conditions* under which learning takes place.

The ability to generalize, of course, is sensitive to the number of hidden units in the network. Thus, if the number of hidden units is too small, the learning problem is over-constrained and there is not enough structure for the network to capture all the needed abstractions; and if the number is too large, the problem is under-constrained and generalization is not possible (Chandrasekaran and Goel, 1988). For this reason, we repeated the above experiment, changing the number of hidden units in the network from 3 to 60 in steps of 3. While the number of games played before the network learned to draw its opponent showed some change with change in the number of hidden units, we observed no apparent change in the *information content* of what was learned.

Computational Intractability of Learning

It has been shown that the learning problem addressed by connectionist networks is computationally intractable (Judd, 1987; Kolen, 1988; Blum and Rivest, 1988). For instance, Kolen (Kolen, 1988) has demonstrated that the connectionist learning problem is computationally intractable by showing that any learning mechanism employed by connectionist networks can also be used to solve the conjunctive-normal form[7] (CNF) satisfiability problem which is known to be *NP-complete*(Cook, 1971)[8].

[7] A conjunctive-normal form expression is a boolean expression comprised of the disjunction of conjunctions of boolean variables. In symbolic form, $\bigwedge_{i=1}^{n} C_i$, where $C_i = \bigvee_{j=1}^{m} x_j$, where x_j is a boolean variable of the negation of a boolean variable. For example $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee \neg x_5) \wedge (x_1 \vee \neg x_4)$ is CNF(Garey and Johnson, 1979).

[8] Nondeterministic polynomial time (NP) is the class of problems where guessing a solution and verifying its correctness can be performed in time bounded by some polynomial function of the input size. Nondeterministic polynomial time complete (NP-complete) is the subset of NP problems such that all problems in NP can be reduced, in polynomial

To illustrate the proof, consider the hypothetical network shown in Figure 7. The network contains fixed and plastic weights such that the network implements the logical functions associated with evaluation of the CNF expression and the selection of a set of satisfying variable assignments. The network has a single input and a single output and four layers of units, where each unit performs a linear threshold operation on its net input. The fourth layer is a single unit performing conjunction of the third layer's output. The third layer implements the individual sets of disjunctions used in the expression. The second layer provides negated values for the disjunctive layer. These three levels, consisting of all fixed weights determined by the given CNF expression, perform the evaluation of the CNF within the network. The first layer, with plastic weights, determines the truth assignments for the variables of the boolean expression according to the weights on that layer. Clearly, this network directly implements the CNF expression when an active input is applied to it. Satisfiability, therefore, is equivalent to finding a set of weights which generate an active output to this input. As CNF satisfiability is known to be NP-Complete, it follows that the problem of learning in linear threshold networks is also NP-Complete. Since an arbitrary network which uses a sigmoid activation function can be simulated by a network of linear threshold units, the above proof also implies that learning is computationally intractable for sigmoid networks as well.

Insert Figure 7 about here.

time, to NP-complete problems. Although not proven, it is generally believed that no polynomial time algorithms exist for problems in NP-complete since the number of possible solutions exponentially with the size of the input.(Garey and Johnson, 1979)

Concluding Discussion

We suggested earlier that an information processing system capable of learning must have several interrelated abilities.

- **Representational Adequacy.** It must be able to represent what it knows and also what it is to learn.
- **Credit Assignment** It must be able to identify its structural components which are responsible for incorrect performance for a specific case of a given task.
- **Structure Modification.** It must be able to change its structure so that it can correctly perform the given task for the specific case.
- **Generalization.** It must be able to generalize (and specialize) what it learns to exemplar-independent abstractions that it needs to perform a given task.
- **Computational Complexity.** It must be computationally efficient so that the learning can occur in a reasonable amount of time.

However, the experiment on learning to compute the XOR function suggests that the computational efficiency of learning in PDP networks depends not as much on the learning procedure as on the choice of initial weights. That is, the efficiency of learning is dependent on the *initial conditions*. Similarly, the experiment on learning to play *Tic-Tac-Toe* suggests that the content of what is learned in PDP networks is often exemplar-specific and not always generalized to exemplar-independent abstractions (prototypes). In fact, the ability to generalize is dependent on the *environmental conditions*. In most connectionist networks, these generalized abstractions are explicitly embedded in the network by the system designer rather that learned by the system. Finally, we have shown that the learning problem as formulated in connectionist networks is NP-Complete.

Potential Solutions

The problems with training in PDP networks described in this paper raise two interrelated issues: what are the causes for these problems, and what are the potential solutions for learning both efficiently and effectively. First, PDP networks, (in fact, neural networks in general) are poor in their capacity for representing knowledge. For instance, McCarthy (McCarthy, 1988) has pointed to the "unary fixation" of connectionist representational schemes, i.e. their apparent inability to easily represent higher-order relations. This representational poverty leads to an incapacity for generalization over abstractions. Recent research on distributed representations (Hinton and Sejnowski, 1986) and recursive representations (Miikkulainen and Dyer, 1988; Pollack, 1988) appears to hold some promise in at least partially alleviating this problem.

Second, PDP networks (again, neural networks in general) typically lack built-in structure (Feldman *et al.*, 1988). For instance, in PDP networks, the structural components responsible for incorrect system performance are "identified" by back propagation of corrective feedback to individual connections in the network, and the system structure is "modified" by adjusting the connection weights in the direction of steepest gradient descent in the error space. These two ideas are captured in the generalized delta rule. However, the generalized delta rule is only a more general, recursive form of hill climbing (Rumelhart *et al.*, 1986a). The problems with the use of the hill climbing technique for navigation of a complex search space which contains local minima are well known (Charniak and McDermott, 1985). What is needed is a decomposition of the learning space so that system can navigate simpler, smaller spaces more efficiently (Chandrasekaran *et al.*, 1988).

The need for decomposing the learning space raises the issue of how to search for the "right" kind of structure. One possiblity is in the direction of developing *task-specific* architectures. For instance, *abductive inference*(inference to the best explanation for a set of data) appears to be ubiquitous in cognition. Recently, several connectionist architectures have been proposed for solving this task (Goel *et al.*, 1988b; Peng and Reggia, 1989; Thagard, 1989). These architectures specify decompositions of the space of explanatory hypotheses that leads to efficient and effective navigation of the underlying problem space.

Acknowledgments

This research has benefited from scholarly contributions of Dean Allemang. This paper benefited from discussions with B. Chandrasekaran, Jordan Pollack, and Tom Bylander. As usual, the authors are responsible for any remaining errors. This work has been supported by the Office of Naval Research, grant N00014-89-J-1200, and by the National Science Foundation, grant CBT-87-03745.

References

A. Blum, R. Rivest, (1988). Training a 3-node Neural Network is NP-Complete. *Proceedings of IEEE Conference on Neural Information Processing Systems*, Denver, Colorado.

J.G. Carbonell, R.S. Michalski, and T.M. Mitchell, (1983). An Overview of Machine Learning. In *Machine Learning: An Artificial Intelligence Approach*, Jaime G. Carbonell, Ryszard S. Michalski, and Tom M. Mitchell (editors), Palo Alto, California: Tioga Publications.

G.A. Carpenter and S. Grossberg, (1987). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics, and Image Processing*, 37:54-115.

B. Chandrasekaran and A. Goel, (1988). From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(3):415-424, May/June 1988.

B. Chandrasekaran, A. Goel, and D. Allemang, (1988). Connectionism and Information Processing Abstractions: The Message Still Counts More Than the Medium. *AI Magazine*, 9(4):24-34, Winter 1988.

E. Charniak, and D. McDermott, (1985). *Introduction to Artificial Intelligence*, Reading MA; Addison-Wesley.

S. A. Cook, (1971). The Complexity of Theorem-Proving Procedures. *Proceedings of the Third ACM Symposium on Theory of Computing*, 1971, pp. 151-158.

J.A. Feldman, M.A. Fanty, N.H. Goddard and K. Lynne, (1988). Computing with Structured Connectionist Networks. *Communications of ACM*, February 1988, pp. 170-187.

S. Franklin, M. Garzon, (1988). Neural Network Implementation of Turing Machines. Poster presentation at the Second IEEE International Conference on Neural Networks, San Diego, California.

M. R. Garey, D. S. Johnson, (1979). *Computers and Intractability*, New York, New York, W H. Freeman and Company.

A. Goel, J. F. Kolen, and D. Allemang, (1988a). Learning in Connectionist Networks: Has the Credit Assignment Problem Been Solved? *Proceedings of the SIGART Aerospace Applications of Artificial Intelligence Conference*, Dayton, Ohio, Vol. II, pp. 74-80.

A. Goel, J. Ramanujam, and P. Sadayappan, (1988b). Towards a Neural Architecture for Abductive Reasoning. In *Proceedings of the Second IEEE International Conference on Neural Networks*, San Diego, California, pp. I:680-688.

G. E. Hinton, and T. J. Sejnowski, (1986). Learning and Relearning in Boltzman Machines. In *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volume 1*, D. Rumelhart, J. McClelland, and the PDP Research Group (editors). Cambridge, MA: MIT Press. A Bradford book. 1986.

J. J. Hopfield, (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings National Academy of Sciences, USA*, 79:2554-2558.

S. Judd, (1987). Learning in Networks is Hard. *Proceedings of the First IEEE International Conference on Neural Networks*. San Diego, California, pp. II:685-692.

J. F. Kolen, (1988). Faster Learning Through a Probabilistic Approximation Algorithm. *Proceedings of the Second IEEE International Conference on Neural Networks*, San Diego, California, pp. I:449-454.

J. McCarthy, (1988). Epistemological Challenges for Connectionism. *Behavioral and Brain Sciences*, 11(1):44.

W. S. McCulloch and Walter Pitts, (1943). A logical calculus of the ideas immanent in neural nets, *Bulletin of Mathematical Biophysics*, Volume 5, pp. 115-137.

Risto Miikkulainen and Michael G. Dyer, (1988). Forming Global Representations With Extended Backpropagation. *Proceedings of the Second IEEE International Conference on Neural Networks*, San Diego, California, pp. I:285-292.

Marvin Minsky and Seymour Papert, (1988). *Perceptrons: An Introduction to Computational Geometry*, Expanded Edition; Cambridge, MA: MIT Press, 1988.

D. Rumelhart, J. McClelland, and the PDP Research Group (editors), (1986a). *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volume 1*, Cambridge, MA: MIT Press. A Bradford book.

D. Rumelhart, J. McClelland, and the PDP Research Group (editors), (1986b). *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volume 2*, Cambridge, MA: MIT Press, A Bradford book, 1986.

D. Rumelhart, G. Hinton, and R. Williams. Learning Internal Representation by Error Propagation. In *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volume 1*, D. Rumelhart, J. McClelland, and the PDP Research Group (editors). Cambridge, MA: MIT Press. A Bradford book. 1986.

D. Rumelhart, P. Smolensky, J. McClelland, and G. Hinton, (1986). Schemata and Sequential Thought Processes in PDP Models. In *Parallel Distributed Processing: Explorations in The Microstructure of Cognition; Volume 2*,D. Rumelhart, J. McClelland, and the PDP Research Group (editors); Cambridge, MA: MIT Press, A Bradford book.

J. Pollack, (1987). On Connectionist Models of Natural Language Processing. Ph.D. Thesis, Computer Science Department, University of Illinois, Urbana.

J. Pollack, (1988). Recursive Auto-Associative Memories: Devising Compositional Distributed Representations. *Proceedings Tenth Annual Conference of the Cognitive Science Society*, Montreal, Canada, August 1988.

Y. Peng and J. Reggia, (1989). A Connectionist Model for Diagnostic Problem Solving. *IEEE Transactions on Systems, Man, and Cybernetics*, in press.

A. Samuel, (1967). Some Studies in Machine Learning Using the Game of Chequers II - Recent Progress. *IBM Journal of Research an Development 11(6):601-617.*

P. Smolensky, (1986), Information Processing in Dynamical Systems: Foundations of Harmony Theory. In *Parallel Distributed Processing: Explorations in The Microstructure of Cognition. Volume 1*,D. Rumelhart, J. McClelland, and the PDP Research Group (editors); Cambridge, MA: MIT Press, A Bradford book.

P. Thagard, (1989). Explanatory Coherence. *Behavioral and Brian Sciences*, in press.

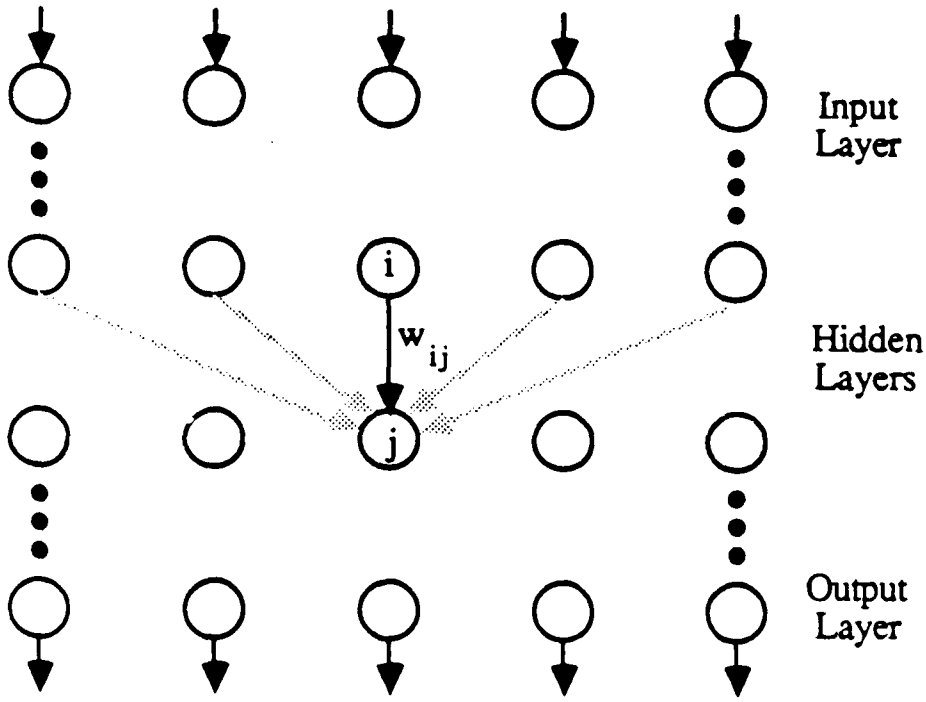

Figure 1: Connection between two processing units

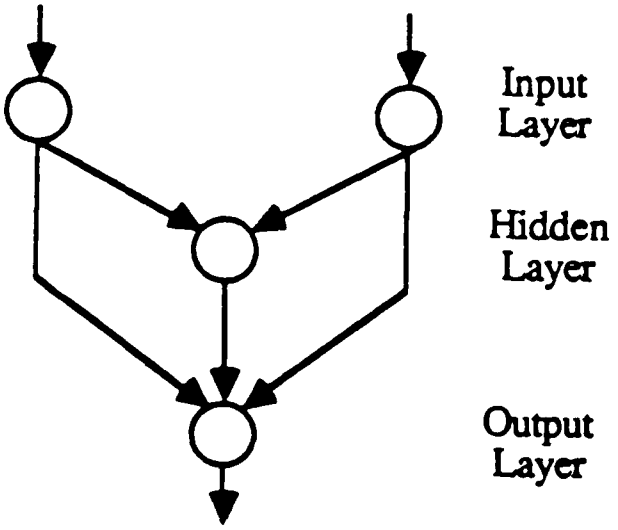

Figure 2: XOR Network

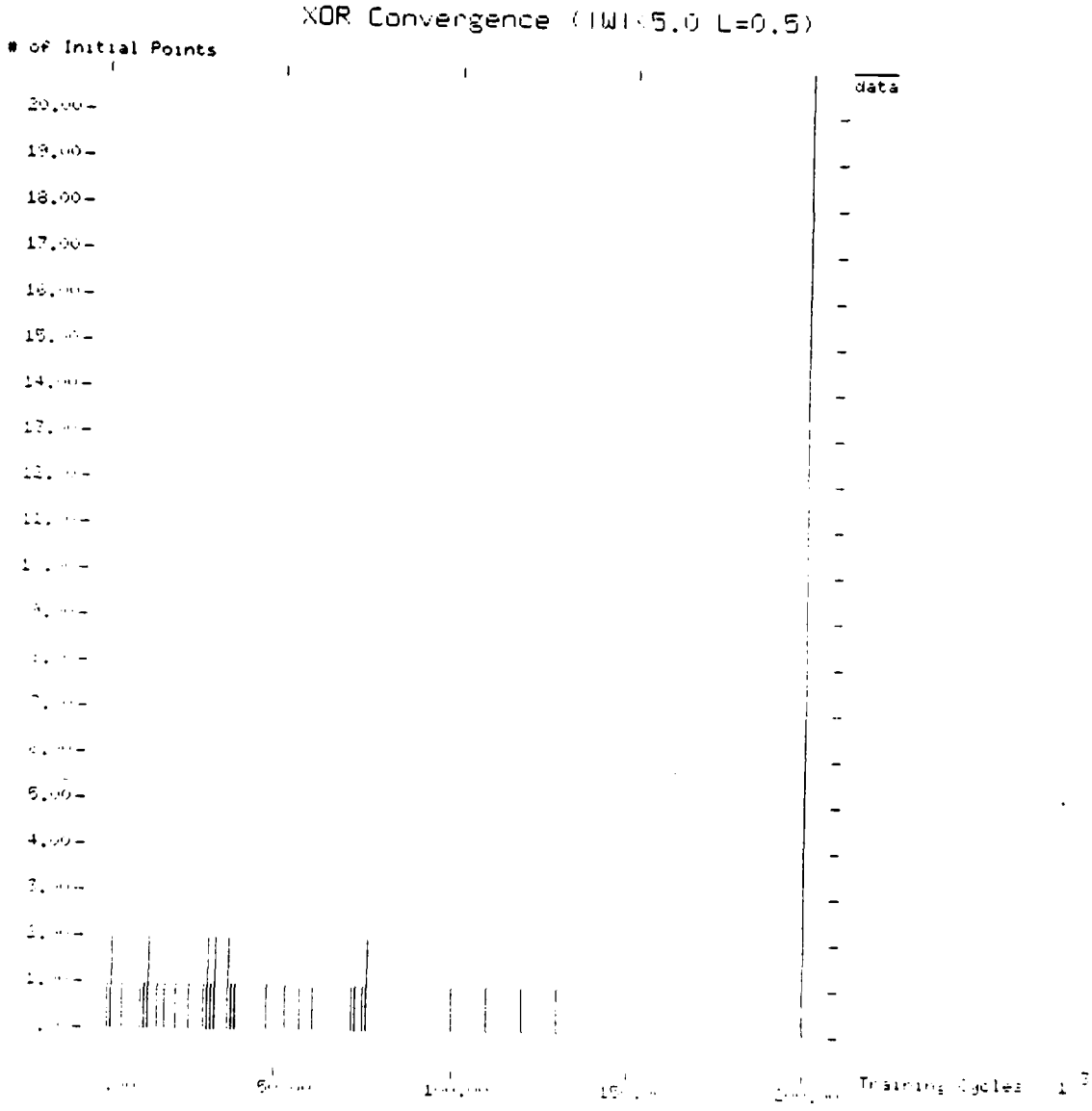

Figure 3: Histogram of XOR Convergence Times (Data Set at 200,000 reduced from 63)

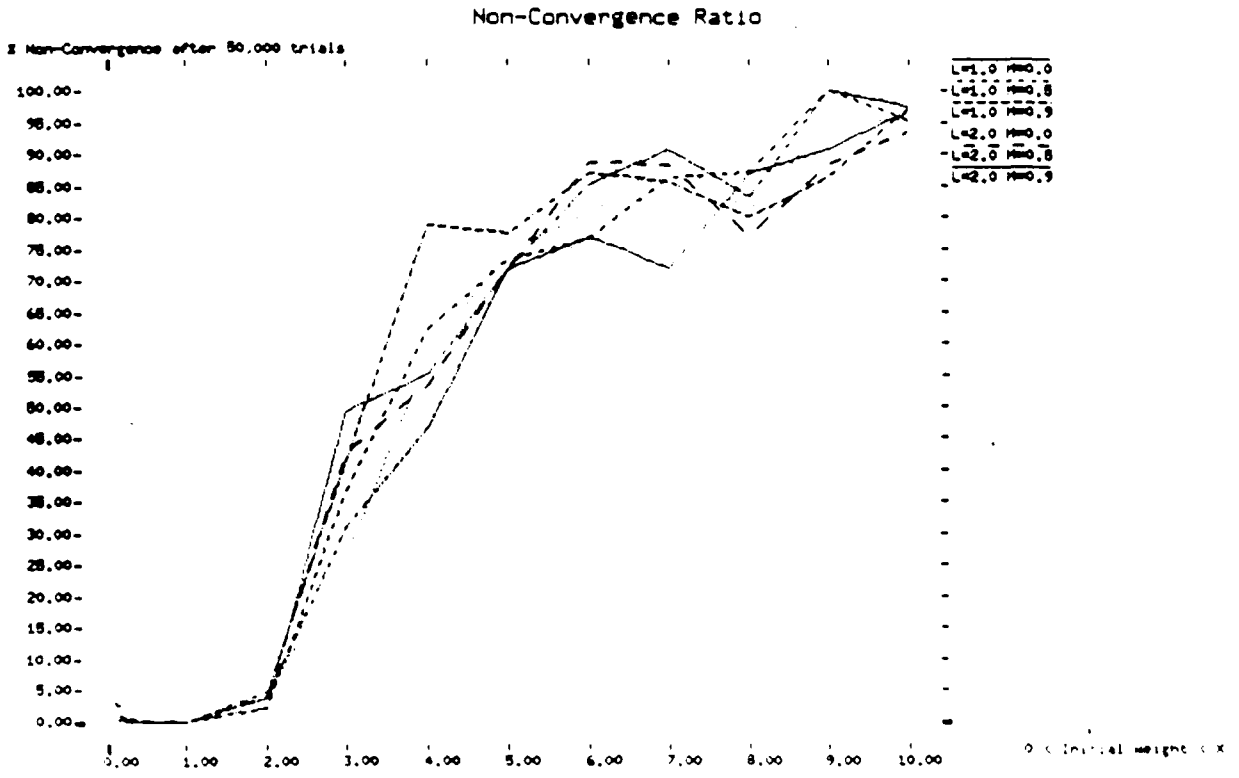

Figure 4: XOR Non-Convergence Percentage versus Initial Weight Range (0.0 to x)

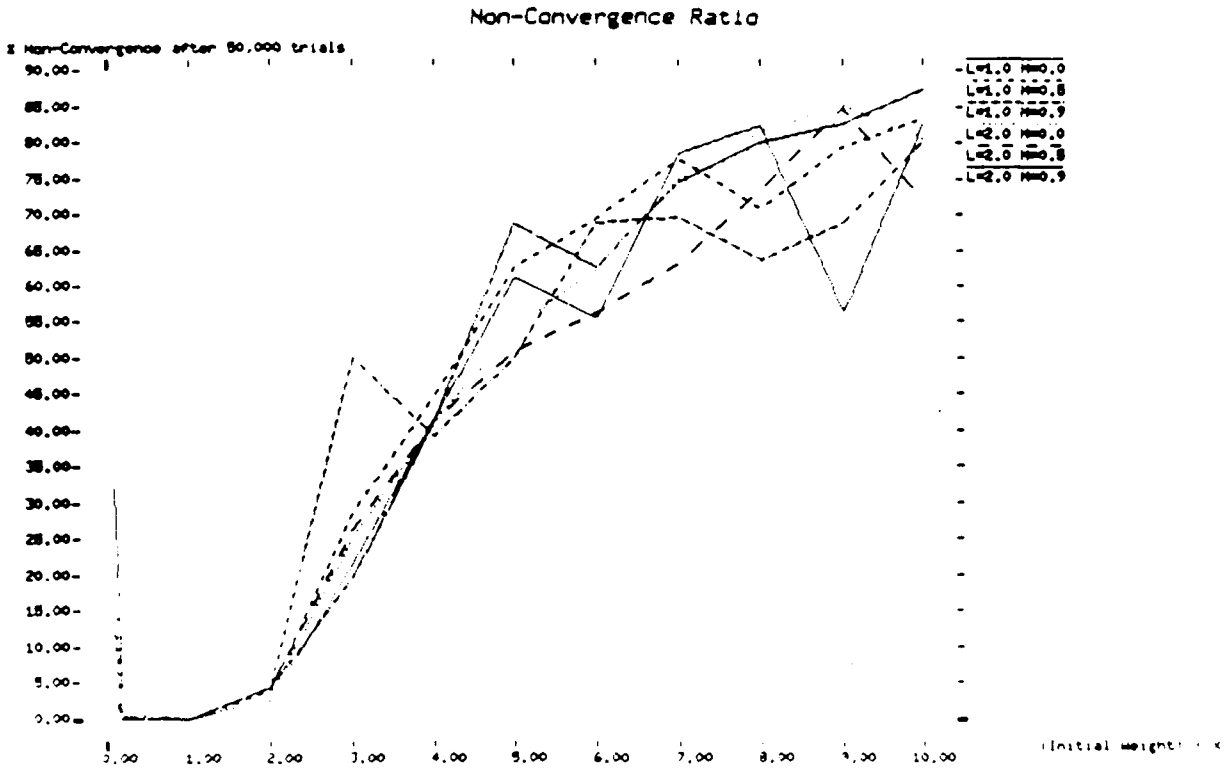

Figure 5: XOR Non-Convergence Percentage versus Initial Weight Range ($-x$ to x)

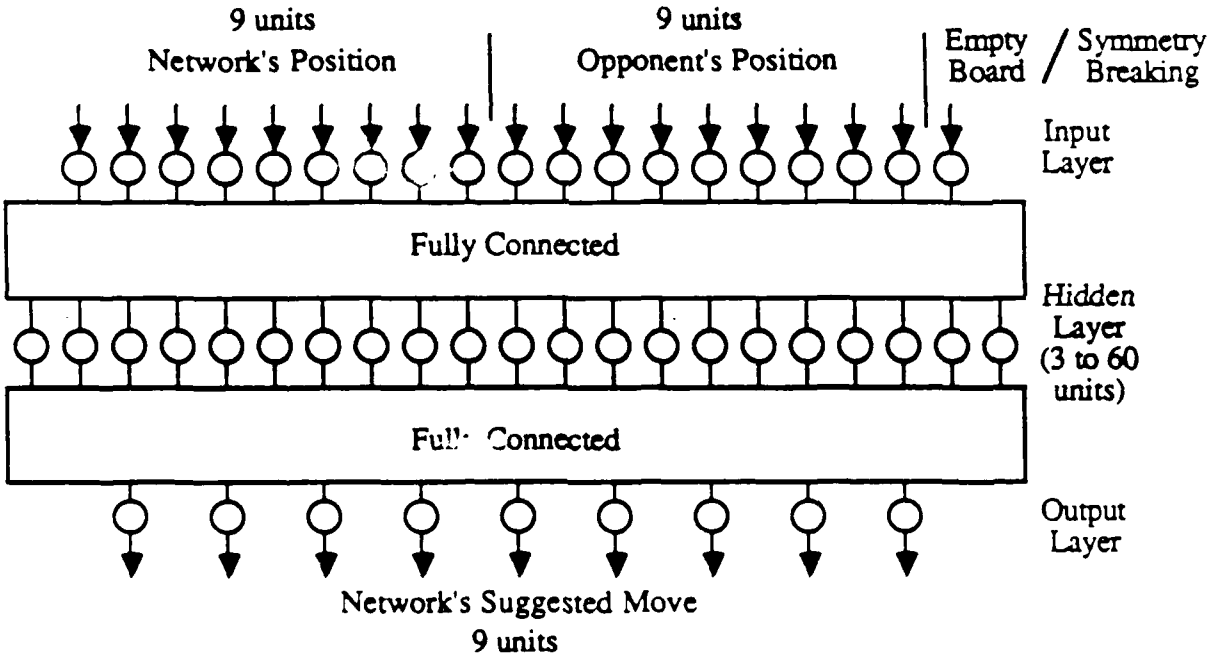

Figure 6: Tic-tac-toe Network

	Symbolic Algorithm Wins (out of 1000)	Neural Network Wins (out of 1000)	Symbolic Algorithm/ Neural Network Draws (out of 1000)
Match 1	33	0	967
Match 2	54	1	945
Match 3	56	0	944
Match 4	57	0	943
Match 5	63	0	937
Match 6	85	0	915
Match 7	277	0	723
Match 8	298	0	702
Match 9	298	0	702
Match 10	345	0	655

Table 1: Tic-Tac-Toe Results

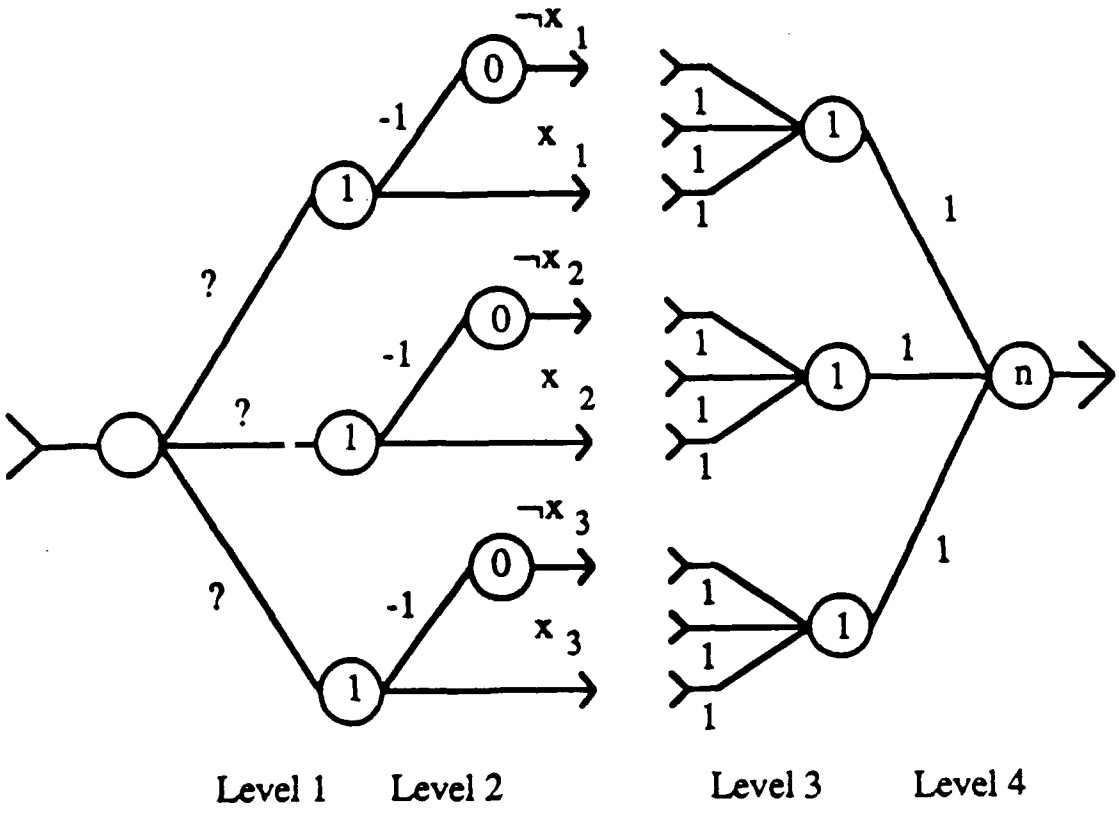

Figure 7: The Generic Conjunctive-Normal Form Expression Network